
**Treball de Fi de Grau**

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

---

# DISSENY, DESENVOLUPAMENT I GESTIÓ D'UN PROJECTE DE SOFTWARE

---

Joaquim Gifre Rosales

Director: Jesús Cerquides Bueno
Realitzat a: Departament de
Matemàtiques i Informàtica

Barcelona, 28 de gener de 2018

## Resum

En aquest treball de fi de grau, es tractarà el desenvolupament actual de productes de Software. Des del moment inicial del treball, es partirà d'una idea i s'analitzarà i treballarà sobre aquesta idea per a acabar tenint un producte de Software funcional.

Aquesta manera de treball, seguint el cicle de desenvolupament de software, farà que el producte de software passi per les següents fases: una fase de observació del problema i planificació, seguida d'una fase d'anàlisi i disseny del problema i el producte, una altra d'implementació del producte amb el seus tests i corresponent integració i una darrera fase de manteniment.

Tenint en compte que es treballarà seguint els principis de les metodologies Agile (en aquest cas concret Scrum) les fases esmentades anteriorment seran iteratives per a cada Sprint.

A més d'optar per una metodologia Agile, s'adoptarà el BDD (desenvolupament dirigit per comportament) com a mètode principal de desenvolupament, en lloc d'altres mètodes com podria ser el TDD (desenvolupament dirigit per tests), cosa que produirà que es pensi sobre el comportament que s'espera dels diferents serveis de l'aplicació abans d'entrar a la fase d'implementació.

Durant el transcurs del projecte, es desenvoluparà una aplicació de Software que constarà de tres parts: un backend conformat per una RestAPI, un frontend desenvolupat amb HTML i Angular5 amb les diferents pàgines per a accedir als diferents serveis del backend i una aplicació Android la qual també disposarà de les suficients pantalles per a fer les crides necessàries al backend.

Aquestes tres parts interactuaran entre elles de tal manera que des de l'aplicació Android s'enviaran dades d'ubicació GPS i temps al backend i aquestes es podran monitoritzar en el frontend.

A més d'aquest sistema de seguiment, es disposarà d'un sistema de gestió i agrupació dels usuaris des del frontend i Android que inclourà tant el registre, el login, la creació de grups i la possibilitat de poder-se unir a grups d'altres usuaris.

Durant el desenvolupament de l'aplicació, assumiré tres papers que cal mencionar:

- El paper de DevOps per a desplegar els serveis per a la Integració Continua (CI). D'entre aquests serveis cal mencionar GitLab, Jenkins i Sonar.
- El paper de Project Manager per a planificar les metodologies de treball i el backlog del producte utilitzant Jira per a mantenir-ne la visió i seguiment durant el desenvolupament.
- Per acabar, el paper de Dev per a, seguint el planificat com a PM, anar desenvolupant els diferents aspectes de l'aplicació seguint la metodologia Scrum i BDD.

## Resumen

En este trabajo de fin de grado, se tratará el desarrollo actual de productos de Software. Desde el momento inicial del trabajo, se partirá de una idea y se analizará y trabajará sobre dicha idea para acabar teniendo un producto de Software funcional.

Esta forma de trabajo, siguiendo el ciclo de desarrollo de software, hará que el producto de software pase por las siguientes fases: una fase de observación del problema y planificación, seguida de una fase de análisis y diseño del problema y el producto, otra de implementación del producto con sus tests y correspondiente integración y una última fase de mantenimiento.

Teniendo en cuenta que se trabajará siguiendo los principios de las metodologías agile (en éste caso concreto Scrum) las fases anteriormente mencionadas se sucederán de forma iterativa para cada Sprint.

Además de optar por una metodologia Agile, se adoptará el BDD (desarrollo dirigido por el comportamiento) como método principal de desarrollo, en vez de usar otras métodos como puede ser el TDD (desarrollo dirigido por tests), lo cual producirá que se piense sobre el comportamiento que se espera de los diferentes servicios de la aplicación antes de entrar a la fase de implementación.

Durante el transcurso del proyecto, se desarrollará una aplicación de Software que constará de tres partes: un Backend formado por una RestAPI, un Frontend desarrollado con HTML y Angular5 con las distintas páginas para acceder a los diferentes servicios del Backend y una aplicación Android la cual también dispondrá de las suficientes pantallas para hacer las llamadas necesarias al Backend.

Estas tres partes interactuarán entre ellas de forma que des de la aplicación Android se enviarán datos de ubicación GPS y tiempo al Backend y dichos datos podrán monitorizarse en el Frontend.

Además de éste sistema de seguimiento, se dispondrá de un sistema de gestión y agrupación de los usuarios desde el Frontend y la aplicación Android que incluirá tanto el registro, el login, la creación de grupos y poderse unir a grupos de otros usuarios.

Durante el desarrollo de la aplicación, asumiré tres papeles que debo mencionar:

- El papel de DevOps para desplegar los servicios para la Integración Continua (CI). Entre estos servicios, debe mencionarse GitLab, Jenkins y Sonar.
- El papel de Project Manager para planificar las metodologías de trabajo y el backlog del producto utilizando Jira para mantener la visión y seguimiento durante el desarrollo.
- El último papel será el de Dev para, siguiendo lo que se haya planificado como PM, ir desarrollando los diferentes aspectos de la aplicación siguiendo la metodología Scrum y BDD.

## Abstract

This final degree project is about the current development of Software products.
From the beginning, this project will take an idea which will be analyzed and which will be developed in order to get a functional Software product.

This way of work, following the Software Development Life Cycle, will make the software product go through the following phases:
One stage about observation of the problem and make the planification, followed by one stage of analysis of the problem and design the product, one stage for the implementation of the product and its tests and its integration and one last stage for maintenance.

Having in mind that the project will be developed following the principles of the Agile methodologies (in this case will be using Scrum) the stages previously mentioned will be repeated for each Sprint.

In addition to choose one Agile methodology, BDD (Behavior Driven Development) will be used as main development methodology instead of using other methods like TDD (Test Driven Development) what will make the developer to think about the expected behavior of all the different services before starting the implementation of the features.

During the course of the project, a Software application that will consist of three parts (one Backend that will be a RestAPI, one Frontend developed with HTML and Angular5 with the different pages to get access to the different services of the Backend and one third part that will be an Android Application with enough screens to make the different calls to the Backend) will be developed.

All these three parts will interact with each other in order to let the Android Application send GPS and time data to the Backend and this data could be visualized from the Frontend. Additionally, the Application will allow the user to register, make login, create groups and join existing groups from other users.

During the development of this application, I will play three different roles:

- DevOps role to deploy and do maintenance of all the services to use Continuous Integration (CI). This CI services include GitLab, Jenkins and Sonar.
- Project Manager (PM) role for planning the different work methodologies and doing the Product Backlog using Jira to keep track of the project during the development.
- Finally, Dev role to develop the application following PM planning using Scrum and BDD.

Voldria dedicar unes paraules d’agraïment a tots els membres de la Unitat de Desenvolupament de Software d’Eurecat per al seu recolzament, paciència i ensenyaments al llarg del temps passat amb ells. Agraïr al tutor per part de la universitat el temps dedicat a corregir i donar consells sobre aquest treball.

Agraïr també al meus pares, familia i amics pel seu recolzament i confiança al llarg dels anys invertits en aquest grau. Agraïr en darrer lloc a la meva parella el seu recolzament i ànim durant els trams més dificultosos i estressants d’aquest camí.

Gràcies a tots.

# Sumari

# Sumari imatges

# Introducció

Durant el transcurs d'aquest treball de fi de grau es desenvoluparà una aplicació de software seguint el cicle de desenvolupament de software. Aquest cicle és una estructura que s'aplica al desenvolupament de productes de software i descriu unes pautes en forma d'etapes que serveixen per al desenvolupament.

En general, les diferents etapes per les quals passa el producte de software es poden agrupar en les tres següents:

1. **Etapa de planificació**: l'objectiu d'aquesta etapa és el reconeixement dels requisits i l'anàlisi del problema que es tracta.
2. **Etapa d'implementació i test**: etapa en la que s'implementa el codi per al projecte i es realitzen els diversos tests amb la finalitat de trobar els errors quan abans millor.
3. **Etapa de desplegament i manteniment:** comença quan el codi ja s'ha probat i es desplega a un entorn de producció. El manteniment ajudarà en la millora i solució d'errors del software un cop desplegat.

S'adoptarà la metodologia Agile Scrum de forma que les diferents etapes es succeiran una vegada per a cada iteració o Sprint del desenvolupament.

Al llarg del projecte és desenvoluparà una aplicació que constarà de tres part que funcionaran de forma conjunta. Les parts a desenvolupar seran les següents:

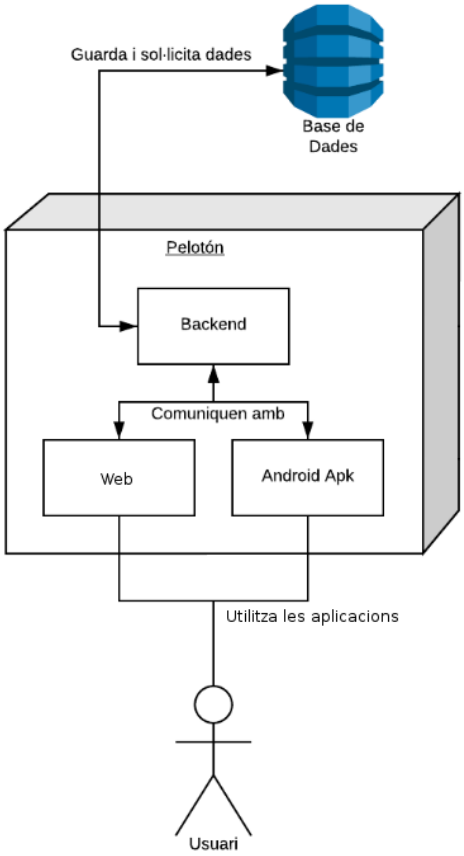
1. **Frontend Web:** part del producte de software amb la que interactuen els usuaris i que serveix com a via d'entrada i de presentació de dades del Backend.
2. **Aplicació Android:** aplicació per a dispositiu smartphone Android que servirà com a Frontend per a dispositius mòbils dels usuaris.
3. **Backend:** part de l'aplicació que processa les dades d'entrada que venen dels Frontend i les retorna per a la seva visualització.

Durant el transcurs del projecte, es treballarà assolint tres papers diferents. Els papers a mencionar són els següents: DevOps, Project Manager i Dev. Fem un petit incís en què representen cadascun dels rols anteriorment esmentats:

- **DevOps:** metodologia amb la que es gestiona el cicle de desenvolupament de software de manera que casa les tasques de desenvolupament i de sistemes. Introduirà mecanismes i eines d'Integració Contínua (IC) per a automatitzar tasques repetitives.
- **Project Manager:** disciplina que engloba l'organització, plantejament, motivació i control de recursos amb la finalitat d'assolir els objectius proposats per a assolir l'èxit. Això suposarà el desenvolupament d'un Backlog i l'organització de les tasques a realitzar al llarg del desenvolupament.

- **Dev:** especialista capaç de concebre i elaborar paquets de software, així com d'implementar-los i mantenir-los utilitzant un o diversos llenguatges de programació. Englobarà el desenvolupament de les tres parts anteriorment esmentades del producte amb els llenguatges pertinents.

El que es planteja amb el treball és la implementació d'una aplicació de Software que permeti als usuaris la creació d'escamots amb un punt de sortida i un d'arribada amb la finalitat d'organitzar quedades ciclistes amb més gent a l'hora de que la seva ubicació es pugui visualitzar.




*Imatge 1 Arquitectura de l'aplicació (I)*

Com es veu en l'anterior diagrama, la única interacció que realitzarà l'usuari és mitjançant el Frontend web i l'aplicació Android. Aquestes dues aplicacions hauran de realitzar peticions al Backend per a, amb les dades introduïdes per l'usuari, realitzar les operacions pertinents i rebre'n el resultat. El Backend serà l'únic que tindrà accés a la base de dades.

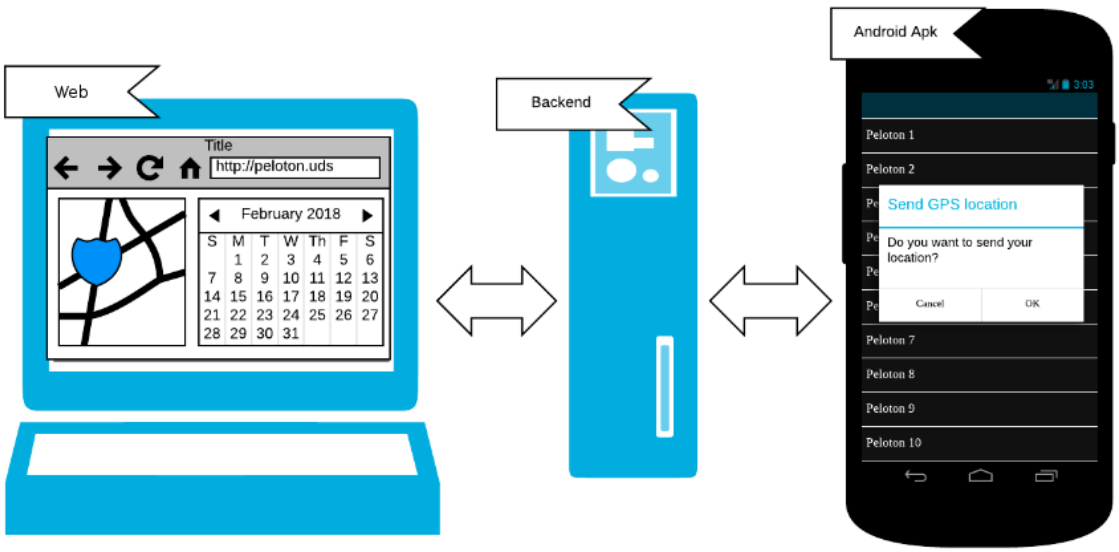
## Objectius

En el treball es realitzarà una aplicació pensada per a facilitar les activitats esportives grupals, en aquest cas les ciclistes, permetent a l'usuari poder crear grups els quals amb un parell de coordenades (una per a indicar-ne el punt de partida i l'altra la de destí) i una hora, els diversos usuaris que s'hi apuntin puguin realitzar la dita activitat esportiva amb més gent.

A més d'aquest component social, es vol que l'activitat de l'usuari quedi enregistrada. Aquesta monitorització consistirà en l'obtenció de les coordenades GPS de l'usuari gràcies a la aplicació per a Android i la data i hora d'aquestes coordenades. Això pot servir tant com per a que un entrenador pugui monitoritzar l'activitat del seu equip o per a que els membres puguin veure on s'ha quedat un membre al que hagin perdut de vista.

Per tal d'assolir-ho, l'aplicació ha de comptar de tres parts diferenciades:

1. Una RestAPI que faci de Backend i permeti la gestió d'usuaris (registre i login a la plataforma), la gestió d'escamots (creació de l'escamot i que s'hi pugui unir un usuari) i la transmissió (recepció i enviament) de paquets de dades que continguin la ubicació de l'usuari.
2. Una Web que serà un Frontend de l'aplicació des del que es pugui accedir a diferents pantalles amb l'objectiu que l'usuari que el faci servir pugui registrar-se, fer el login, crear i unir-se a escamots i fer el seguiment dels usuaris en diversos escamots.
3. Una aplicació per a Android que contingui les suficients pantalles per a que l'usuari pugui registrar-se, fer login, crear i unir-se a escamots i l'enviament de dades d'ubicació.



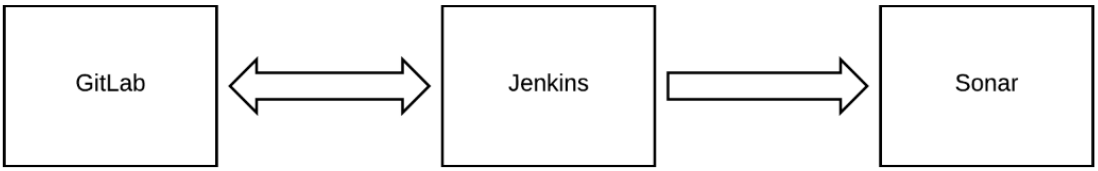

*Imatge 2 Arquitectura de l'aplicació (II)*

S'espera que tant la web com l'aplicació per a Android únicament es comuniquin amb el Backend. Aquest darrer serà el que realitzarà les accions necessàries i l'únic que es comunicarà amb la base de dades.

Què s'espera de cadascuna de les diferents funcionalitats?

- **Registre:** per a aquesta acció es vol que l'usuari introdueixi les dades necessàries per a fer el seu registre a la plataforma. Aquestes dades seran: nom, cognoms, email, Dni, data de naixement, username i password.
- **Login:** es vol que gràcies a la introducció del nom d'usuari i el password es garanteixi l'accés a les restants accions a l'usuari. Si no ha realitzat el login, l'usuari no ha de ser capaç de realitzar més opcions que no siguin el login o registrar-se.
- **Creació d'un escamot:** acció per la qual es vol que l'usuari crei un escamot. La creació de l'escamot implicarà la introducció d'un nom per a l'escamot, una hora d'inici, la latitud i longitud d'inici, la latitud i longitud de final de recorregut i el nombre màxim de membres que es vol per escamot. L'usuari que crei un escamot, se'n converteix automàticament en organitzador.
- **Unir-se a un escamot:** acció per la qual es vol que un usuari s'afegeixi a la llista de membres de l'escamot. Aquesta acció es realitzarà sempre i quan l'escamot no estigui plè.
- **Enviar les dades GPS:** es vol que l'usuari pugui enviar al Backend la seva latitud i longitud, data i hora d'aquesta presa de dades. Únicament es mantindrà en base de dades un objecte coordenada, representant de la darrera coordenada coneguda de cada usuari.
- **Rebre dades GPS:** es recuperaran les últimes coordenades conegudes de, o bé, un usuari en concret, o bé, de tots els membres de l'escamot.

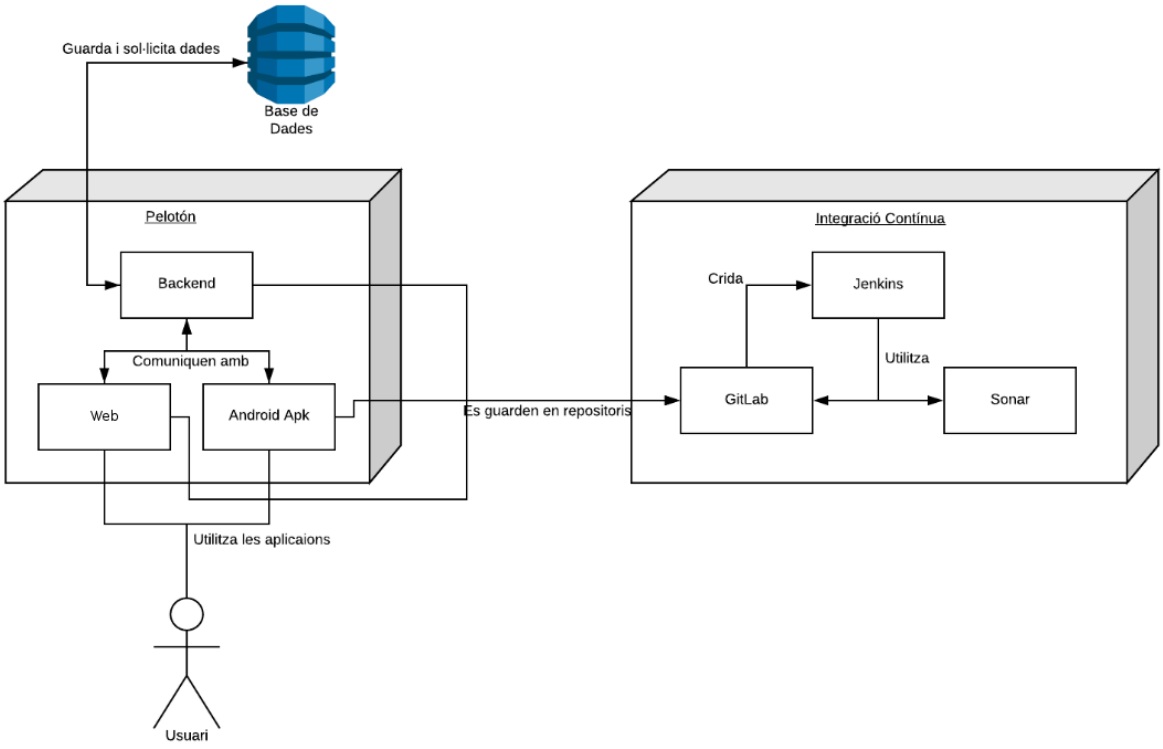
Paral·lelament al desenvolupament de l'aplicació, es vol realitzar el desplegament en una màquina virtual externa de diverses eines per a l'aplicació d'integració contínua. Aquestes eines seran Jenkins, GitLab i Sonar i es vol que interactuïn de la següent manera:




*Imatge 3 Serveis d'integració contínua*

Amb aquest darrer esquema s'il·lustra les comunicacions entre els diversos serveis de IC ja que volem el següent:

- Volem que GitLab allotgi en diversos repositoris cadascuna de les parts de l'aplicació a desenvolupar.
- Volem que Jenkins escolti si hi ha cap event push a GitLab per a executar el job corresponent. També volem que Jenkins pugui clonar el repositori i executar-ne el Gradle, la qual cosa suposarà, a més de la compilació, l'execució dels tests per finalment fer una crida a Sonar.
- Volem que Sonar executi un anàlisi de qualitat sobre la part que ha desencadenat el job de Jenkins.




*Imatge 4 Visió global del projecte*

Veiem en aquest esquema que es poden distingir dues parts en el projecte:

- Una primera part a l'esquerra la qual és l'aplicació que es vol desenvolupar. En aquests aplicació, l'usuari interactua amb els Frontends i aquests amb el Backend.
- Una segona part a la dreta la qual és l'entorn d'integració contínua. Aquest entorn ajudarà en el control de versions gràcies a GitLab i en exercir un control de qualitat en l'aplicació amb l'ajuda de Sonar. Jenkins s'encarregarà d'executar els jobs i veure que el resultat és una build estable.

A més de l'aplicació i l'ús d'integració contínua, s'espera que el treball es desenvolupi seguint unes metodologies de treball concretes, sent més precís les metodologies Scrum i BDD. Scrum enfocarà la planificació del treball al llarg del procés mentre que BDD enfocarà el desenvolupament del Backend.

# Desenvolupament

## Eines, tecnologies i metodologies utilitzades

Durant el transcurs del treball, s'han utilitzat diverses tecnologies i metodologies de treball. En quant a tecnologies utilitzades, s'han necessitat eines per a la realització de les següents tasques:

- Per a mantenir un control de versions de l'aplicació s'ha utilitzat GitLab.
- Per a mantenir un control dels bugs que apareguin en el desenvolupament s'ha utilitzat Sonar.
- Per a mantenir el sistema d'integració contínua s'ha utilitzat Jenkins.
- Per a mantenir l'anàlisi i planificació del projecte s'ha utilitzat JIRA.
- Per al desenvolupament del Backend s'utilitzarà Spring Boot.
- Per al desenvolupament de la Web s'utilitzarà Angular5 a partir d'Angular CLI.
- Per al desenvolupament de l'aplicació Android s'utilitzarà Android Studio.
- Per a realitzar un primer test del Backend s'utilitzarà Postman.

Per la banda de les metodologies, trobem que s'ha utilitzat una metodologia per al desenvolupament global al llarg del pojecte i l'altre durant el desenvolupament del Backend. D'aquesta manera tenim:

- Scrum com a metodologia utilitzada durant tot el transcurs del projecte amb la finalitat de mantenir un anàlisi i planificació global de la feina a realitzar.
- BDD com a metodologia de treball durant la implementació del Backend.

A continuació s'introduiran les diverses eines i metodologies esmentades agrupades en diverses seccions. Aquestes, correspondran al tipus de rol el qual està destinada aquella metodologia o tecnologia i, això vol dir que, es parlarà d'eines per a la Integració Contínua, eines i metodologies com a Project Manager i per últim eines i metodologies com a Dev.

## Eines per a la IC

Les primeres eines sobre les que s'hauria de parlar són les que s'han utilitzat per al desenvolupament com a DevOps. La integració continua és una pràctica mitjançant la qual es realitzen integracions automàtiques d'un projecte utilitzant una eina per a control de versions per a poder executar probes de forma automàtica i detectar errors de forma ràpida.

Les eines utilitzades, en aquest cas, amb aquesta finalitat són: Gitlab, Jenkins i SonarQube.

### GitLab


*Imatge 5: Logo GitLab*

**Què és GitLab?**

Un servei web que serveix per a realitzar control de versions i el desenvolupament col·laboratiu amb el projecte de software basat en Git.

El control de versions és la gestió dels diversos canvis que es realitzen sobre els elements del producte de software.  Una versió, doncs, és l'estat en el que es troba en un moment donat del seu desenvolupament.

El fet de tenir un control de versió facilita l'administració de les diferents versions del producte donat la facilitat de veure o recuperar les modificacions realitzades en cada control.

La tecnologia es basa en un mecanisme d'emmagatzematge dels elements que conformaran el producte, la possibilitat de realitzar canvis en aquests elements guardats i un històric que registra les accions realitzades amb cada element que s'hi guarda.

**Per què s'ha escollit?**


*Imatge 6 Comparativa serveis Git*

Al mercat hi ha diversos serveis que serveixen per a realitzar la mateixa funció. D'entre les alternatives podem trobar GitHub o Bitbucket. Si analitzem similituds entre característiques dels productes, podem trobar les següents:

- Pull request.
- Code review.
- Inline editor.
- Issue tracking.
- Suport de llenguatge Markdown.
- Autenticació a dos passos.
- Administració de permisos avançats.
- Pàgines web allotjades estàticament.
- APIs pròpies amb nombroses prestacions.
- Possibilitat de fer fork o clonar repositoris.
- Snippets.
- Integracions 3rd party.

Tot i que amb les esmentades similituds es podria pensar que qualsevol de les tres opcions és vàlida per al desenvolupament del projecte, tot i això, hi ha diferències significatives entre els tres.

Si ens fixem en GitHub, trobem un servei el qual:

- Restringeix la creació de repositoris privats de manera que solament els poden utilitzar usuaris que paguen quota mensual. El preu varia en funció a la quantitat de projectes privats que es poden crear fins a un màxim de 50 repositoris.
- Té restriccions en quant a la mida de fitxers i de repositori. En quant a fitxer es restringeix la mida a fins 100MB i fins a 1GB.

Si mirem Bitbucket trobem el següent:

- El nombre de repositoris privats és il·limitat però hi ha restricció en quant a la quantitat de col·laboradors que hi pot haver en el repositori. La mínima quantitat (i gratuïta) és de 5 usuaris.

GitLab també té el seu sistema de pagament per a aconseguir habilitar un seguit de les característiques esmentades anteriorment. Per altra banda, és un sistema open-source en el que l'usuari s'encarrega d'instal·lar el servei i configurar-lo al seu gust, òbviament dins de les capacitats facilitades segons si ha pagat o no, amb la possibilitat de generar repositoris privats il·limitats per a cada usuari, permetre la gestió i agrupació dels usuaris i altres característiques dins les quals podem trobar, per exemple, un sistema de correu electrònic per defecte (GitLab Rails) el qual envia correus als usuaris o un Ngnix per a aixecar el servidor on mantenir el GitLab.

Jenkins

*Imatge 7 Logo Jenkins*

**Què és Jenkins?**
Jenkins és una aplicació per a la integració i desplegament continuu. Serveix per a fer builds i poder córrer els tests sobre la aplicació de software que s'estigui desenvolupant de forma continua. D'aquesta manera es facilita la tasca als desenvolupadors a l'hora d'integrar els canvis al projecte i també als usuaris donat que facilita l'obtenció de builds recents.

La manera de treballar de Jenkins és mitjançant la definició i creació de pipelines que continguin les pautes de comportament de Jenkins (a cadascún dels pipelines se l'hi pot programar l'execució de diferents eines en seqüència) i les diferents eines de test que es vulgui que utilitzi sobre l'aplicació.

**Per què s'ha escollit?**
Hi ha al mercat diversos programes que  com poden ser Travis CI o Codeship que realitzen també les tasques de integració i desplegament continuu.

Travis CI és una opció fàcilment descartable degut que funciona amb GitHub i que funciona amb i per a aplicacions open-source i degut a l'elecció de GitLab per a crear el repositori privat de forma gratuïta, en lloc d'aprofitar el servidor de GitHub, descartem aquest servei.

L'altra opció esmentada és la de Codeship. Aquesta, en la seva versió gratuïta, ens posa a l'abast un seguit de màquines pre-configurades amb diferents dependències de integració continua ja instal·lades i és fàcilment programable gràcies a una interfície gràfica en la web.Tot i això, la quantitat de builds que es poden executar de forma simultània i les pipeline de test que s'executen de forma paral·lela està limitada a un.

Jenkins el podem desplegar en una màquina o en un servidor que nosaltres decidim. A més de ser una instal·lació senzilla a partir d'un *.war, tenim també amb Jenkins la possibilitat de configurar l'entorn gràcies a una interfície gràfica a la que podem accedir fàcilment a partir d'un navegador.

Totes les eines que necessitéssim utilitzar amb Jenkins són fàcilment instal·lables gràcies a un sistema de plugins que es poden instal·lar i mantenir al dia via la GUI mencionada anteriorment.

L'última cosa a esmentar sobre l'elecció de Jenkins és que resulta ser un sistema gratuït íntegrament que podem fer servir per a la integració i desplegament continuat per a fer builds o testejar l'aplicació en els múltiples sistemes operatius OS X, Linux o Windows.

SonarQube


*Imatge 8 Logo SonarQube*

**Què és Sonar?**
SonarQube és una aplicació open-source que serveix per a mesurar i analitzar la qualitat de codi. L'aplicació pot ser instal·lada de forma local i mitjançant l'eina complementaria anomenada Sonar Runner es poden realitzar els anàlisis des de la pròpia màquina on s'hagi instal·lat.

Tot i la possibilitat de realitzar un anàlisi en local quan l'usuari ho requereixi, el punt fort radica en la possibilitat d'incloure aquesta eina en processos i eines de integració contínua de manera que al generar-se una nova build s'executi un anàlisi.

Sonar realitza aquest anàlisi en base a uns paràmetres de qualitat que l'usuari pot configurar i és capaç de trobar bugs, punts vulnerables, fragments de codi repetits, entre altres. A més, dóna una valoració general sobre el codi que ha analitzat i, segons els criteris donats, et diu si ha fallat en quan a superar la "nota de tall".

Degut a que està pensat per a treballar de forma conjunta amb eines d'integració contínua, el programa permet veure les diferents puntuacions en les diferents branques que tingui el repositori.

A més d'analitzar per branques, cada troballa que faci de possible errada, bug o vulnerabilitat és mostrada com a issue i et facilita la informació de la seva ubicació en el codi i una explicació dels motius de per a quin motiu ha estat marcat.

Com a últim detall, Sonar és també una eina que conta amb una GUI accessible des del navegador i de fàcil accés i gestió la qual permet, a més de la visualització dels projectes que analitza i els resultats, configurar el servei i afegir-hi fins i tot diversos plugins per a ampliar-ne les funcionalitats.

**Per què s'ha escollit?**
Analitzant el mercat, trobem que hi ha més opcions en quant a programes o serveis els quals puguin analitzar codi i retornar un informe de qualitat. D'entre les alternatives sobre les que m'he informat es troben serveis com pot ser FindBugs o Codacy.

Podem veure que FindBugs també és un aplicació open-source la qual també disposa d'una GUI accessible des d'un navegador i que dóna els anàlisis amb la ubicació i motiu de les errades detectades, tot i que l'explicació del motiu no és tant extensa com pot ser amb Jenkins per exemple. Aquest programa té una pega, més aviat limitació, molt gran i és que només serveix per a analitzar aplicacions realitzades amb Java.

Tenint en compte que durant el projecte volem fer un Backend o l'aplicació Android programant en Java, podría ser d'utilitat. Tot i que pensar d'aquesta manera implicaria buscar una altra eina per a analitzar, si calgués, el Frontend programat en un llenguatge diferent a Java, en el nostre cas Angular5 utilitzant Typescript i HTML. Es descarta doncs FindBugs degut a la limitació a un únic llenguatge per al seu anàlisi.

L'altre eina mencionada, Codacy, és una eina més complerta d'anàlisi de codi que proporciona els seus informes al detall com pot fer Jenkins i que té també la seva pròpia GUI per a la visualització dels processos que realitza.

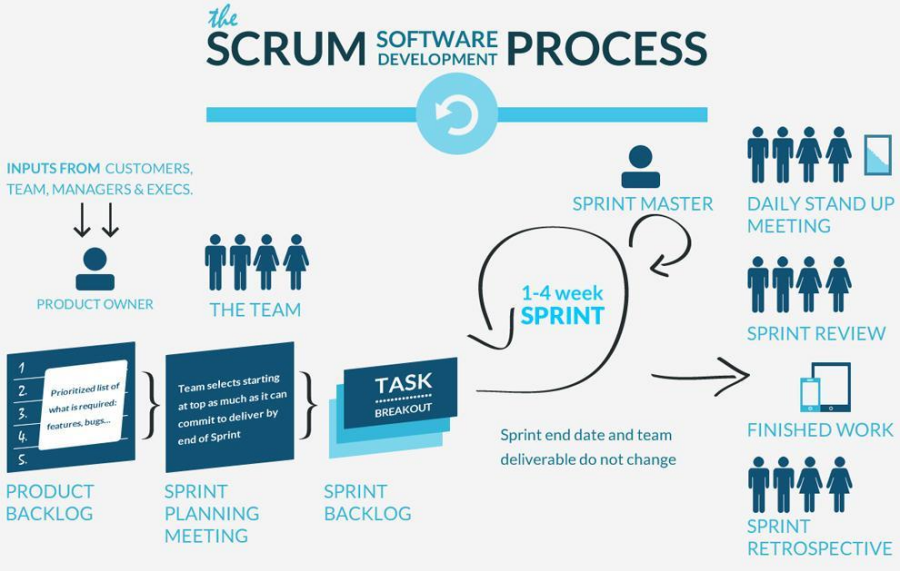
Tot i les bondats que pugui tenir, es pot veure que no és una eina open-source i que està allotjada a un servidor propi. Quines pegues podem trobar-li si es veu tant bé a simple vista? Doncs que es basa també en un sistema de pagament per a poder utilitzar al complert les seves funcions i utilitzar la versió gratuïta implica l'ús de GitHub o Bitbucket. Les limitacions de GitHub amb els repositoris privats o de Bitbucket amb el nombre de col·laboradors fan que acabi sent una opció descartable.

## Eines i metodologies com a Project Manager

A continuació s'explica tot el que, durant el transcurs del projecte, ha estat relacionat amb el desenvolupament de tasques com a Project Manager (PM per a abreviar).

Al parlar dels recursos utilitzats durant el projecte realitzant el paper de PM s'ha de parlar d'una metodologia de treball i d'una eina. En quant a la metodologia utilitzada per a realitzar les tasques, s'ha optat per utilitzar la metodologia Agile Scrum i la eina per a realitzar-ne el seguiment ha estat el JIRA d'Atlassian.

### Scrum




*Imatge 9 Diagrama de la metodologia Scrum*

**Què és Scrum?**

Scrum primerament és una metodologia Agile i, això, vol dir que:

1. És una estratègia de desenvolupament incremental i iteratiu en el que les diverses fases de planificació, anàlisi, disseny, implementació i test es repetiran a cada iteració.
2. Els requisits i les solucions poden evolucionar al llarg del temps segons les necessitats del projecte. Això implica que es doni prioritat a la flexibilitat per a adaptar-se al canvi.
3. Els equips que treballin en el projecte estaran auto-organitzats i seran multidisciplinaris de manera que tots interactuaran en el procés de presa de decisions a curt termini.

A més a més, Scrum defineix un conjunt de pràctiques, rols i ítems els quals es poden prendre com a punt de partida a l'hora de definir el procés de desenvolupament.

Els rols que descriu són:

- **Product Owner:** persona que escriu les User Stories que formaran part del Product Backlog i les valora, prioritza i gestiona.

- **Scrum Master:** s'assegura que el procés de Scrum s'utilitza i transcorre sense incidents. Ha d'eliminar obstacles que impedeixin que l'equip arribi a l'objectiu del Sprint.
- **Team:** equip petit amb les habilitats necessàries per a realitzar la feina. Té la responsabilitat d'implementar i lliurar el producte al final de cada Sprint.

Les pràctiques descrites són:

- **Sprint:** així és com es denomina a cada iteració. És un període de curta duració, entre 1 i 4 setmanes que ha de finalitzar amb un prototip operatiu o un producte potencialment entregable.
- **Sprint Planning Meeting:** reunió al principi del Sprint en la que els diferents membres de l'equip decidiràn la feina a fer.
- **Daily Meeting:** reunions diàries breus realitzades de forma àgil, fins i tot de peu, per a que no s'allarguin amb el simple objectiu d'informar del que es va fer ahir, el que es farà avui i els problemes que s'hagin trobat.
- **Sprint Review:** reunió realitzada al final del Sprint en la que es tracta la feina que s'ha pogut completar i quina no. Es mostrà també la feina feta al Product Owner.
- **Sprint Retrospective:** també es realitza al final de Sprint i serveix per a que els implicats en l'Sprint donin les seves impresions per a millorar, si escau, el procés.

Per últim, els ítems que descriu Scrum són:

- **User Stories:** descripció curta i fàcil de recordar d'una funcionalitat o requisit que serà útil per a l'usuari. A més de la funcionalitat han de tenir la seva "definition of done", és a dir, han de contenir una manera de comprovar la seva implementació.
- **Product Backlog:** document per a tot el projecte el qual inclou tots els requisits del projecte. El contingut és el recull de totes les històries d'usuari (user stories) prioritzades segons una escala de valors.
- **Sprint Backlog:** subconjunt de requisits pertanyents al Product Backlog que han de ser desenvolupats durant el següent Sprint. Les tasques que hi ha al Sprint Backlog són distribuïdes pels membres de l'equip de la manera que els sembli més adient.
- **Burn down chart:** representació gràfica de la feina realitzada en el projecte durant un sprint. Usualment a l'eix vertical hi ha la valoració de la feina a realitzar i en l'horitzontal el temps.

**Per què s'ha escollit?**

Hi ha diverses metodologies Agile d'entre les quals es pot destacar Kanban o eXtreme Programming. Si ens fixem en aquestes dues, de forma breu podríem dir el següent:

- **Kanban:** metodologia Agile basada en la idea que, la feina en curs (WIP - Work In Progress) s'ha de limitar i només s'ha de començar amb una tasca nova quan el bloc de feina anterior o bé, ja ha estat entregat, o bé, ha passat a una funció posterior de la

cadena. Entenent per cadena un esquema com pot ser: TODO -> WIP -> Test -> DONE.

- **eXtreme Programming (XP):** metodologia Agile centrada en potenciar les relacions interpersonals com a clau de l'èxit per a desenvolupar el producte de software. Intenta promoure el treball en equip procurant l'aprenentatge i un bon ambient de treball per a cada membre de l'equip.

Tenint en compte que el treball de fi de grau ha estat realitzat en solitari, la idea de treballar utilitzant XP ha resultat senzilla de descartar.

La idea de treballar en Kanban és del tot acceptable i es pot utilitzar perfectament per a desenvolupar un projecte com el que es realitza en aquest treball.

Tot i això, gràcies a que les metodologies Agile no són rígides, la metodologia utilitzada durant aquest treball ha estat la combinació de Scrum i Kanban, la qual ha acabat generant un seguit de documentació al llarg del desenvolupament i ha organitzat tot el procés.

JIRA


*Imatge 10 Logo JIRA*

**Què és JIRA?**
JIRA és una eina online que serveix per a la administració de tasques d'un projecte, el seguiment d'errors i incidències i la gestió operativa de projectes. Aquesta eina permet:

- Una planificació del projecte mitjançant la creació d'històries d'usuari i incidències. També es poden crear Sprints i distribuir les diferents tasques entre els diferents membres de l'equip.
- Supervisar i analitzar el treball de l'equip sencer en el seu context i amb completa visibilitat.
- Crear informes amb les dades recollides durant el transcurs dels Sprint. D'aquesta manera, s'automatitza per exemple la generació de burn down charts.

**Per què s'ha escollit?**
Al mercat trobem també diverses eines que realitzen feines similars o iguals a JIRA. Podem trobar per exemple Blossom, Countersoft Gemini o Trello.

D'entre les quatre aplicacions esmentades, JIRA, Blossom i Gemini son tres aplicacions configurables al gust de l'usuari molt complertes que permeten tant la planificació com la visualització del projecte que es vol desenvolupar a l'hora que ofereix eines per a mantenir connexió amb altres serveis com poden ser de repositoris o de missatgeria per a notificar als membres de l'equip de la situació del projecte si alguna cosa canvia.

El contra principal per a qualsevol d'aquests tres serveis és que no són serveis gratuïts i no disposen d'una versió limitada per a un únic usuari. En aquest punt és on s'introdueix Trello el qual si bé és cert que disposa d'una versió de pagament, es pot utilitzar de forma gratuïta.

Trello és un servei que proporciona eines per a la planificació del projecte mitjançant taules Kanban personalitzables i té a l'abast un ventall de plugins que n'amplien les funcionalitats per a, per exemple, poder utilitzar-lo per a treballar amb Scrum o eines d'anàlisi per a dibuixar burn down charts.

És en aquest punt en el que es pot trobar la pega en Trello. Si bé és cert que disposa d'una gran quantitat de plugins que poden ampliar bonament les funcionalitats bàsiques que ofereix la plataforma, no tots els plugins disponibles són gratuïts.

En vistes del comentat, tot i haver vist plugins per Trello gratuïts per al desenvolupament en Scrum, he decidit la utilització de JIRA com a plataforma en la que mantenir la planificació i anàlisi durant el transcurs del projecte. Això ha estat degut al fet de que, en l'empresa en la qual estic desenvolupant el treball, disposen de JIRA i allà mantenen seguiment dels seus projectes.

## Eines per al desenvolupament

Les últimes eines i metodologies restants a debatre, són les que s'han emprat en la implementació de l'aplicació. Durant aquesta implementació s'ha adoptat una metodologia de treball anomenada Behaviour Driven Development (BDD), la traducció del qual és Desenvolupament Guiat per Comportament. Les eines utilitzades han estat: SpringBoot, Postman, Angular-CLI i Android Studio.

### Spring Boot


*Imatge 11 Logo SpringBoot*

**Què és Spring Boot?**

Spring Boot és un framework que alleugera la tasca de configuració d'aplicacions basades en Spring. Spring és un framework que utilitza Java per al desenvolupament d'aplicacions web i que utilitza injecció de dependències. Es dirigeix a la administració del cicle de vida dels components Java (coneguts com a Beans) mitjançant la utilització de XML, anotacions i JavaConfig.

Degut al gran nombre de característiques englobades dins Spring o de les dependències que s'hi poden carregar per a ampliar-les encara més, la complexitat a l'hora de configurar Spring resulta en una tasca tediosa i propensa a tenir errors.

Els objectius principals de Spring Boot inclouen proveir als usuaris d'una eina més ràpida i accessible per al desenvolupament amb Spring i oferir un ventall de característiques que són comunes en projectes que abarquin diversos aspectes com poden ser la seguretat, configuracions externalitzades, servidors integrats, etc.

Spring Boot no genera codi i no requereix de configuració XML per al seu funcionament. La mica de configuració que s'ha de realitzar és fa a partir d'anotacions. A més, tot i que Spring Boot ofereix una configuració per defecte, la qual facilita l'execució de l'aplicació, aquesta configuració pot ser modificada per l'usuari.

**Per què s'ha escollit?**

Buscant informació sobre possibles alternatives, podem veure que hi ha altres frameworks com podrien ser ActFramework o Dropwizard els quals podrien considerar-se alternatives a Spring Boot.

Si bé podem trobar informació sobre les virtuds de ActFramework, estem parlant d'un framework la documentació oficial del qual es troba encara en desenvolupament i el qual té

una comunitat força reduïda (de moment durant el transcurs del treball) ha estat descartat aquest framework degut a la insuficient informació que es pot trobar actualment.

Per l'altra part, podem veure que Dropwizard és una altra opció la qual pot realitzar funcions similars a Spring amb una única diferència. La possibilitat de configurar al gust l'aplicació és perd degut a que el framework per si sol s'encarrega de tots aquests aspectes reduïnt, en aquest aspecte, la llibertat del desenvolupador.

Pels motius de tenir una gran comunitat en la que pots trobar la informació necessària tant en la documentació oficial com en alternatives i a la possibilitat de configuració al gust que ofereix Spring, he optat per l'ús de Spring Boot com a framework a utilitzar per al desenvolupament de la RestAPI.

## BDD (Behaviour Driven Development)

**Què és BDD?**
BDD es tracta d'una metodologia de treball la qual ha evolucionat a partir del TDD (Test Driven Development) i que es basa en l'escriptura dels tests abans de començar amb cap mena d'implementació.

La diferència amb entre aquestes dues metodologies de desenvolupament es troba en que, en lloc de pensar i escriure els tests unitaris propis per al TDD, escriurem uns tests que s'encarregaran de comprovar que el codi actui com s'espera des del punt de vista del model de negoci, és a dir, comprovar que el comportament de la funcionalitat sigui l'esperat.

Un cop s'hagin escrit els diversos tests, al igual que en el TDD, es pot començar a implementar la funcionalitat en qüestió. Com és d'esperar, el codi s'anirà polint fins que la funcionalitat que ha estat implementada passi tots els tests.

Per a escriure aquestes probes, s'utilitza un llenguatge específic el qual permet la escriptura igual per a totes les probes. Aquest llenguatge s'anomena Gherkin i s'utilitza per a permetre que qualsevol persona pugui entendre les funcionalitats que es desenvoluparan.

Gherkin és un llenguatge que pot ser entès tant per persones com per màquines degut a que és un llenguatge senzill d'escriure, llegir i entendre. Aquest llenguatge utilitza paraules clau per a poder ser entès per les màquines. Les paraules clau bàsiques són:

- **Feature:** utilitzada per a posar títol a la funcionalitat. El títol, sol anar seguit d'una breu descripció la qual pot ser una història d'usuari.
- **Scenario:** s'utilitza per a introduir l'escenari el qual es vol testejar. Cada escenari inclou com a mínim un Given, un When i un Then.
- **Given:** dona context a l'escenari que es testeja. Es pot entendre com les precondicions que preparen l'escenari a ser testejat.

- **When:** especifica les accions que utilitzarà el test. S'entén com a la interacció que realitza un usuari.
- **Then:** dóna el resultat que s'espera del test. Comprovem si els canvis produïts en el sistema són els que s'esperen.

Gherkin treballa amb una eina anomenada Cucumber la qual s'encarrega d'executar els tests de forma automàtica interpretant el llenguatge Gherkin i fent d'enllaç amb l'aplicació per a executar els passos necessaris.

Per a poder fer d'enllaç, Cucumber utilitza unes expresions regulars per tal de relacionar cada pas del Feature amb la definició de la funció en els step definitions.

**Per què s'ha escollit?**
Tot i mantenir relació amb altres mètodes com pot ser TDD, en relació a escriure primer els tests i després implementar les funcionalitats en qüestió, veig més interessant el fet de no només testejar que una funció de codi retorni el valor que s'espera sinó que es testeja que, de forma global, el resultat a una acció desemboca a la conseqüència esperada, és a dir, el programa es comporta com s'espera.

Postman

*Imatge 12 Logo Postman*

**Què és Postman?**
Postman és una eina la qual permet provar RESTful APIs. Permet, mitjançant una senzilla GUI realitzar peticions HTML sense la necessitat d'escriure codi per a testejar les diferents funcions de la API.

Permet fer els diferents tipus de petició (GET, POST, PUT, PATCH, DELETE, ...) podent-hi afegir diferents capçaleres, body i paràmetres entre altres. A més, permet agrupar i guardar les peticions en col·leccions per a poder-hi accedir fàcilment i tenir-les agrupades.

**Per què s'ha escollit?**
A més de Postman, es poden trobar altres eines que serveixen al mateix propòsit. Algunes de les que es poden trobar si es busquen alternatives al mercat, és poden trobar Insomnia Rest Client o Paw.

Paw és una opció fàcilment descartable donat a que no té una versió gratuïta i a que és una aplicació disponible per a Mac. Si mirem Insomnia, veiem que és una aplicació la qual té una versió gratuïta i versions de pagament les quals ofereixen alguna funcionalitat extra al ventall de possibilitats.

Tot i això, es pot veure que les funcionalitats de les tres aplicacions esmentades són similars per no dir gairebé idèntiques. Així doncs, que Postman sigui una aplicació multiplataforma gratuïta fa que sigui la millor de les opcions.

Angular-CLI


*Imatge 13 Logo Angular CLI*

**Què és Angular CLI?**
Angular CLI és un intèrpret de línia de comandes d'Angular 2 en endavant el qual facilita la creació de projectes realitzats en Angular. Angular és una plataforma que simplifica la creació d'aplicacions web que podran ser visualitzades en diversos dispositius.

Dins l'ecosistema d'Angular trobem la eina anomenada Angular CLI. És una eina oferta pel mateix equip d'Angular la qual ens facilita la creació d'un nou projecte ja que ofereix l'esquelet d'arxius i carpetes que es necessitaran de mateixa manera que diferents dependències ja configurades.

**Per què s'ha escollit?**
En aquest cas no he trobat eines alternatives per a la creació d'una aplicació utilitzant Angular així que, al contrari que amb altres tecnologies que disposaven d'alternatives en les que poder pensar quina utilitzar, per a treballar amb la versió més nova de Angular, la 5 en aquest cas, he decidit la inclusió de Angular CLI per a simplificar la generació i manteniment de codi.

Android Studio


*Imatge 14 Logo Android Studio*

**Què és Android Studio?**

Android Studio és un entorn de desenvolupament integrat (IDE) oficial per a la plataforma Android. És un entorn gratuït el qual a més d'un editor de codis i diverses eines per a desenvolupador que tenen altres editors, ofereix diverses eines que aumenten la productivitat durant la compilació d'aplicacions per Android. Algunes de les millores són les següents:

- El sistema de compilació es basa en Gradle.
- Emulador integrat amb diverses funcionalitats.
- Un entorn unificat per a treballar per a tots els dispositius Android.
- La opció de fer tornar a executar el programa per aplicar canvis mentre ja s'està executant evitant compilar tota la aplicació de nou.
- Eines per a detectar problemes de rendiment, usabilitat, compatibilitat, etc

**Per què s'ha escollit?**

Per al desenvolupament amb Android es poden utilitzar altres eines com podrien ser IntelliJ IDEA o Eclipse. Ambdues d'aquestes eines poden utilitzar-se també per a la creació d'aplicacions Android sempre i quan s'instal·lin els plugins, SDK i altres eines necessàries per al desenvolupament d'aplicacions en Android.

Els motius pels quals s'ha triat Android Studio són sumament senzills:

- És un entorn de desenvolupament gratuït amb tot el necessari per a començar a treballar.
- És un entorn de desenvolupament oficial ofert per Google.

## Explicació del desenvolupament

Arribats a aquest punt, podriem resumir la feina a realitzar en aquest treball amb els següents punts:

- Desenvolupar una aplicació consistent de Backend, Frontend web i una aplicació per a dispositius Android.
- Seguir el model de integració contínua al llarg del projecte utilitzant Jenkins, GitLab i Sonar.
- Seguir una planificació de desenvolupament basada en diverses metodologies de treball.

La qüestió és: per on s'hauria de començar a abordar aquest projecte? Fent memòria, a la introducció d'aquest document es menciona que el desenvolupament de l'aplicació de Software es realitzarà a l'hora que es personifiquen tres rols els quals són presents en el desenvolupament actual de Software: DevOps, Project Manager i Dev.

Podríem començar a desenvolupar indiscriminadament assumint el paper de Dev, és a dir, començant a programar però ens toparíem amb diversos obstacles o preguntes al intentar-ho:

- Si el projecte es basa en IC però encara no tenim les eines preparades, quan se suposa que es començarà a implementar la integració contínua?
- S'ha descrit amb anterioritat algunes funcionalitats que es volen per a l'aplicació però encara no s'ha ni analitzat ni planificat a cap nivell les tasques a realitzar.

Veient que acabaríem programant a cegues i sense tenir tot l'entorn de treball preparat, s'aparta de moment la idea de començar amb el desenvolupament i es comença a treballar realitzant una altra tasca.

S'ha discernir entre si començar planificant o preparant l'entorn de treball utilitzant IC. Arribats a aquest punt, ambdues opcions es podrien considerar gairebé igual d'apropiades. Ara bé, l'entorn per a la integració contínua ha de ser instal·lat de zero en una màquina facilitada per la empresa i, degut a la manca d'experiència en realitzar aquesta mena de tasques, la possibilitat d'oferir-ne un anàlisi i planificació anticipat es presenta com una idea atractiva.

Després d'aquest petit raonament, es pot veure que un dels primers passos lògics a seguir és l'assumpció del paper com a Project Manager per a, en un principi, realitzar un primer Backlog molt global en el que es puguin visualitzar les tasques a realitzar.

## Project Manager

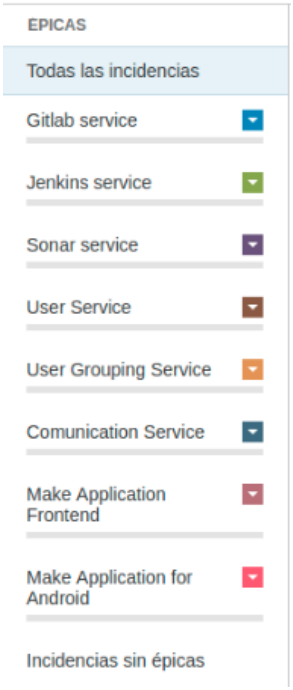
Recordem que el Backlog és una llista de les característiques que l'equip de Scrum ha de desenvolupar per a poder considerar el projecte com a complert. És el principal punt d'entrada de requeriments i queda a disposició com a documentació del projecte, a més, és possible que canvii durant el transcurs del projecte de manera que convé que es mantingui actualitzat per a que compleixi el seu propòsit de forma adequada.

Com s'ha esmentat amb anterioritat, el desenvolupament de l'aplicació es realitzarà seguint la metodologia Scrum i l'escriptura del Backlog ens servirà per a dotar de visió global el projecte anunciant les tasques que s'hauran de realitzar i contribuint a la planificació dels Sprints.

Abans de seguir aprofundint en el desenvolupament, toca introduir unes breus definicions sobre aspectes que engloben Scrum i el Backlog. Aquestes definicions són:

- **Èpiques:** així s'anomenen a aquelles històries les quals impliquen tal quantitat de treball que podrien ser desglossades en històries més petites. Això passa degut a que una Èpica és una història molt genèrica.
- **Història d'usuari:** són la unitat més petita de treball i es coneixen també com a tasques. Aquestes històries narren característiques que seran incloses en l'aplicació en forma d'historieta. Haurien d'incloure uns paràmetres d'acceptació per a que els desenvolupadors puguin saber quan poden considerar la tasca com a realitzada.
- **Versió:** producte de software resultant al cap de diversos Sprints el qual ha de ser suficientment funcional per a ser entregat i que el client el pugui provar.
- **Sprint:** iteració que dura entre 1 i 4 setmanes en la que l'equip de desenvolupament realitza certes tasques seleccionades del Backlog.

La visió global del projecte es realitzarà, doncs, mitjançant l'escriptura d'històries èpiques. Aquestes històries, tot i ser genèriques, donen una pauta sobre la feina que s'haurà de realitzar tot i que, abans de poder implementar res, aquestes èpiques s'han de desglossar en les diverses tasques corresponents al compliment de la èpica.




*Imatge 15 Llista d'èpiques extreta del JIRA*

Tal com es pot veure en la anterior figura, s'han definit com a èpiques 3 tasques per a desplegar els serveis de IC, 3 tasques més per als tres serveis elementals que es volen desenvolupar a nivell de Backend i 2 tasques més, l'una per al Frontend web de l'aplicació i l'altra per a l'aplicació Android.

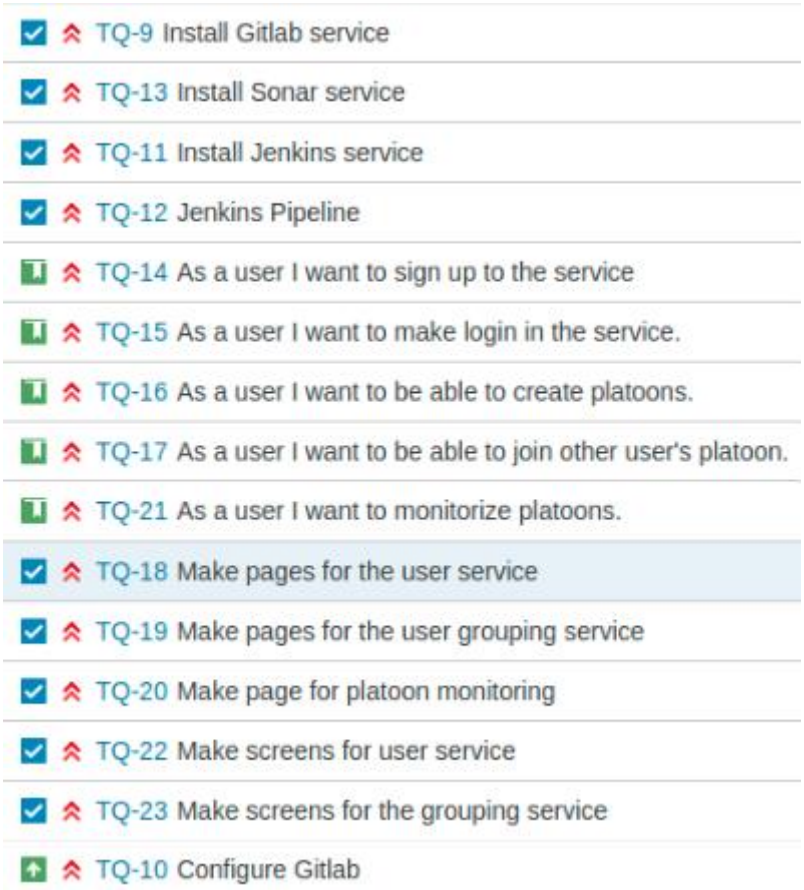
El que esperem referent a les diverses èpiques és el següent:

- **GitLab service:** aquesta tasca ha d'incloure la instal·lació del servei en la màquina virtual esmentada, configurar-ne ports i serveis i creació de comptes i repositoris necessàris per al desenvolupament.
- **Jenkins service:** englobarà també la instal·lació en la màquina virtual de l'aplicació, la seva configuració i la creació de pipelines necessaris per al seu ús.
- **Sonar service:** inclou tant la instal·lació d'aquesta aplicació en la màquina virtual facilitada com la configuració necessària per a que funcioni adequadament.
- **User service:** servei del Backend el qual haurà de oferir a l'usuari el necessari per a que es pugui registrar en el servei i iniciar la seva sessió al servei amb el compte registrat.
- **User grouping service:** servei que oferirà a l'usuari les funcions necessàries per a que pugui crear els escamots o que es pugui unir als que ja hagi creat un altre usuari.
- **Comunication service:** servei que s'encarregarà tant de l'actualització en base de dades de la informació de la última ubicació coneguda de l'usuari com de la obtenció d'aquesta dada per part del Frontend web o Android.
- **Make application Frontend:** tasca la qual es refereix al desenvolupament de tot el que requereixi la realització del Frontend web, és a dir, totes les pàgines i funcions que ajudin a l'usuari en la utilització dels diferents serveis oferts pel Backend.

- **Make application for Android:** tasca que implica la realització de l'aplicació per a Android amb totes les Activities i funcions necessàries per a la utilització de les diverses funcionalitats del Backend.

Ara que ja es té un breu esquema sobre el que s'ha de realitzar, aquestes èpiques s'han d'esmicolar de manera que quedin reduïdes a tasques menys generals i es pugui començar a treballar en el projecte.

Les èpiques, un cop simplificades, conformen un llistat de tasques i històries d'usuari que són les següents:




*Imatge 16 Backlog inicial extret del JIRA*

El següent pas a seguir, ha estat la definició de la durada dels Sprints ja que la valoració de les diferents tasques podria arribar variar depenent del temps que es vulgui emprar per a la realització d'un grup concret de tasques.

Per a fer la estimació de les diferents tasques, s'han fet servir punts d'història. Aquests s'han basat en l'estimació del temps que es tardaria des que s'agafa la tasca fins que es donaria per acabada. Per a decidir-ne els valors, he utilitzat el que es coneix com a Planning póker amb el tutor assignat a l'empresa de pràctiques per, mitjançant un concens, arribar a valorar les diverses tasques.

Tenint en compte que el treballar utilitzant Scrum ajuda al desenvolupament de documentació al llarg del projecte, he decidit que la durada de cada Sprint per a aquestes

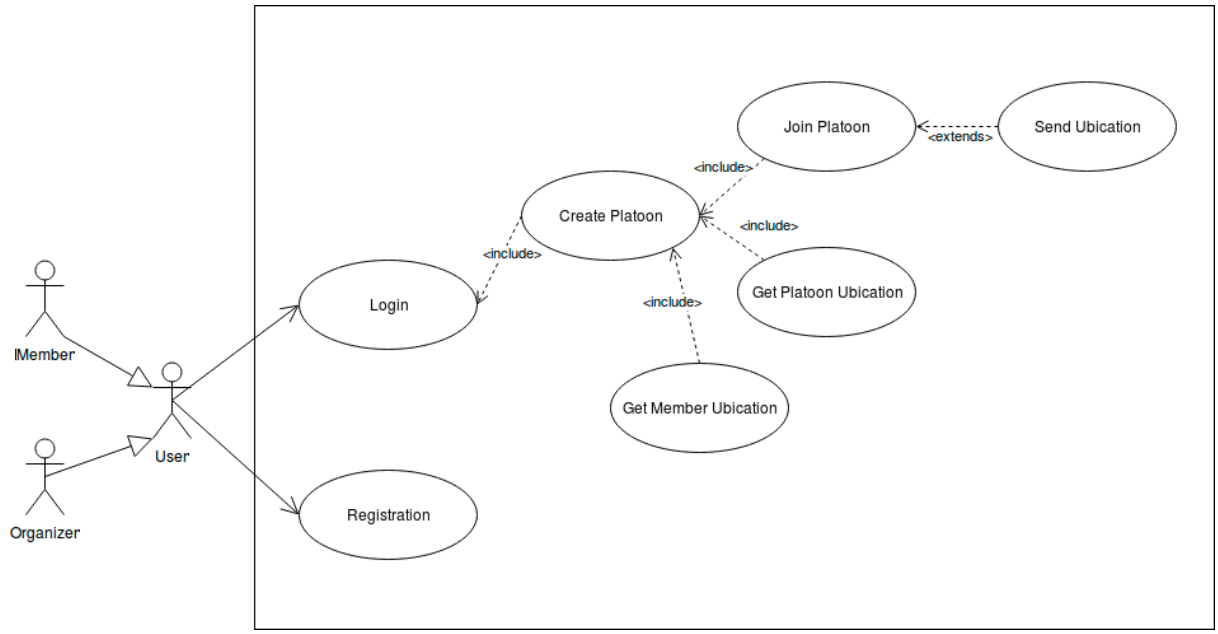
jornades de treball i el temps restant fins al lliurament del projecte serà d'una setmana per a Sprint.

A partir d'aquest punt, es pot dir que el paper de Project Manager quedarà reduït ja a únicament mantenir el Backlog actualitzat, iniciar i tancar els Sprints i analitzar-ne els resultats.

## Anàlisi del producte

Un cop conclòs l'anàlisi de les tasques que haurem de fer per l'aplicació, tenim en mans un producte el qual, tal com s'ha descrit al principi, constarà de tres parts: un Backend, una web i una aplicació per a Android.

Donant un cop d'ull global a l'aplicació, es pot dibuixar el següent diagrama de casos d'ús en el qual es veuen les diverses funcions de l'aplicació:




*Imatge 17 Diagrama de casos d'us*

Què podem observar en l'anterior diagrama? Podem observar tant els diferents papers que adopta l'usuari com les diverses accions que pot emprendre amb l'aplicació.

L'usuari pot ser considerat com a organitzador o com a membre de l'escamot. L'única diferència en aquesta consideració és que l'organitzador és aquell usuari que ha creat l'escamot i un membre és aquell que s'hi uneix.

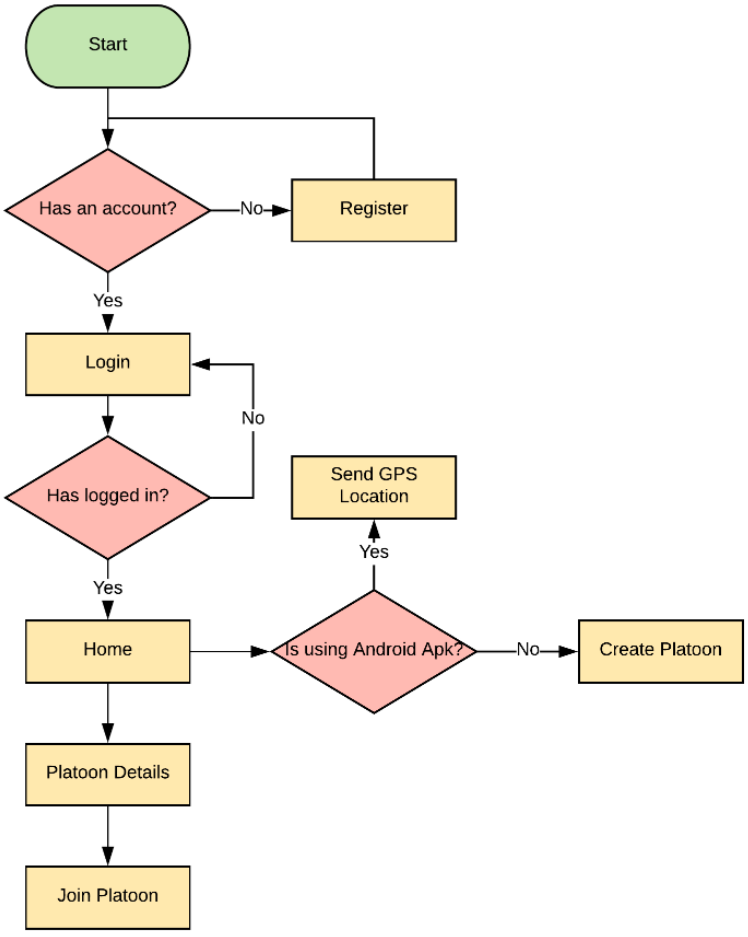
Les úniques opcions que l'usuari pot utilitzar de l'aplicació sense dependre de cap altre acció són la de fer login o la de registrar-se a la plataforma. Les altres accions dependran de que l'usuari hagi pogut realitzar login.

Hi ha una altra dependència, doncs l'única acció que es pot emprendre una vegada s'ha fet login és la creació d'un escamot. La resta d'accions depenen de que com a mínim hi hagi un escamot registrat a la plataforma.

Qualsevol usuari pot visualitzar l'estat de l'escamot i dels usuaris que hi estan apuntats. Tot i això, aquesta característica queda reservada a aquelles persones que estan accedint a l'aplicació via web.

La possibilitat d'enviar la ubicació de l'usuari queda restringida a aquells usuaris que es trobin utilitzant la aplicació des d'un dispositiu Android. Amb aquestes dades s'envia tant la latitud i la longitud de l'usuari com la hora i data d'aquest enviament.

Si observem el flux de l'aplicació, veiem que les diverses pantalles i funcions transcorren de la següent manera:




*Imatge 18 Diagrama de flux de les aplicacions*

Com es pot veure, no totes les funcions poden ser utilitzades en qualsevol plataforma. La creació d'escamots queda restringida a l'us de la web mentre que la ubicació de l'usuari només es pot enviar a través de l'aplicació mòbil. Ambdues decisions han estat preses durant el desenvolupament de l'aplicació Android.

La primera, degut a l'opinió de que, si s'espera que una persona es responsabilitzi de la organització i monitorització de l'activitat, no tindria per raó de ser la possibilitat d'organitzar una activitat de forma esporàdica.

La segona, degut a que s'espera que els usuaris que utilitzin aquesta aplicació amb la finalitat de que un tercer el pugui observar durant l'activitat esportiva no tindria sentit que l'usuari que qui monitoritzi l'activitat es trobi enviant la seva ubicació.

## IC (Integració Contínua)

Amb aquest primer anàlisi ja acabat, el que vaig proposar per a seguir treballant va ser el desplegament dels diferents serveis de integració contínua ja que s'espera que els diferents entorns utilitzats s'utilitzin al llarg de tot el desenvolupament del projecte. Aquest desplegament consistirà en la instal·lació i configuració dels diferents components.

La instal·lació de Jenkins, Sonar i GitLab a una màquina virtual externa implica el traspàs mitjançant una connexió ssh dels fitxers necessaris i la connexió ssh mitjançant la consola per a executar els diversos fitxers d'instal·lació. S'ha de contemplar que per a poder instal·lar aquestes eines, és necessària la instal·lació d'una versió de Java, en aquest cas s'ha utilitzat la de Oracle, i la instal·lació de postgreSQL o MySQL.

El principal problema que pot sorgir en aquest procés d'instal·lació és el solapament de ports dels diferents components. Tot i que per defecte Jenkins s'instal·la al port 8080 i Sonar al 9000, GitLab instal·la diversos components amb els que pot haver-hi problema de solapament concretament, des de la meva experiència, amb Jenkins.

Aquests conflictes s'han d'arreglar editant uns fitxers que contenen les configuracions de les diverses eines i es troben, per descomptat, a la màquina virtual. Això vol dir que per a la edició d'aquests fitxers hem d'utilitzar un editor de text des de consola, en aquest cas, s'ha utilitzat vi.

GitLab s'ha utilitzat per a allotjar tres repositoris. En cadascun d'aquests repositoris s'hi ha allotjat una de les tres parts de l'aplicació, és a dir, s'ha creat un repositori per al Backend, un altre per al Frontend web i un altre per a l'aplicació Android. Cada repositori conté un fitxer .gitignore corresponent per tal de no mantenir en ell més fitxers dels necessàris per a l'aplicació.

La decisió d'allotjar cada part de l'aplicació en repositoris separats en lloc de en un de sol ve donada pel fet que, d'estar tots els fitxers en el mateix repositori, l'ús de les altres eines de IC, com pot ser el Sonar, ens interessa que realitzin anàlisis concrets per a cadascuna de les parts i, en el cas del Jenkins, volem que ens retorni els informes de cada build per a cada

component per separat i que els temps d'execució dels pipelines sigui el més reduït possible degut a només executar la pipeline de la part que es modifiqui.

En el cas de Sonar, s'han de configurar els barems i paràmetres de qualitat a part d'instal·lar els plugins necessàris per a realitzar els diversos anàlisis. En el meu cas, he decidit deixar els paràmetres de qualitat per defecte i he instal·lat únicament un plugin per a possibilitar el testeig d'aplicacions Android.

Un cop fet això, no resta res més a fer amb Sonar degut a que la creació de projectes la realitzarà Jenkins, de ser necessària, al fer una build. Considerarà necessari crear-lo quan s'executi per primera vegada l'anàlisi del projecte i el passi per Sonar. Un cop creat i analitzat el projecte, tots els canvis aniran afectant al projecte ja existent.

Sonar s'utilitzarà, un cop realitzat els anàlisis, per a veure l'estat de qualitat del codi i s'haurà de refactoritzar el codi de tal manera que Sonar indiqui que el codi no té ni bugs ni vulnerabilitats. Les repeticions de codi i "code smells" s'hauran de mantenir al mínim possible.

Per últim, toca parlar sobre Jenkins. Primer de tot, s'ha procedit a instal·lar-lo en la màquina virtual i, un cop feta la instal·lació principal, s'han instal·lat des de la GUI de Jenkins tots els plugins complementaris per a poder cridar a Sonar i GitLab durant l'execució dels jobs.

Primer, fem un petit incís a la terminologia que s'utilitzarà al parlar de Jenkins:

- **Job/Project:** es refereix a les tasques executables que estan controlades i monitoritzades per Jenkins.
- **Build:** resultat de l'execució d'un job.
- **Publisher:** part del procés d'execució a part de la compilació. Cada publisher pot reportar que la build sigui estable o inestable depenent els resultats del seu procés. L'execució dels tests, l'anàlisi del Sonar, el clone realitzat del repositori de GitLab són exemples de publishers.
- **Build estable:** build realitzada amb èxit i considerada estable perquè cap publisher l'ha etiquetada d'inestable.
- **Build inestable:** build considera inestable per un o més publishers tot i que la build s'hagi realitzat amb èxit.
- **Broken/Failed Build:** quan la build falla en el procés de construcció.

Un cop preparades les eines necessàries per a que Jenkins realitzi totes les tasques que utilitzarà cada cop que volem que realitzi una build, toca plantejar quants jobs convé crear. Tot i que el projecte consti de 3 parts, s'ha decidit la creació únicament de dos jobs. Un d'ells s'encarregarà del Backend i l'altre de l'aplicació Android.

He decidit la inclusió de només dos jobs degut a que el tercer, el corresponent al Frontend web, només interactua amb el Backend i mostra les dades. D'aquesta manera, no considero tant necessària la creació d'un job que compili l'HTML i en busqui els errors.

En els dos jobs a definir, s'inclourà el següent en els diversos apartats:

1. **General:** nom per al projecte seguit del tipus de connexió que es vol amb GitLab, en aquest cas GitLab VM.
2. **Source Code Management:** es selecciona aquí el tipus de repositori i es facilita tant la URL del repositori i credencials per a accedir-hi com la branca que s'ha de construir.
3. **Build Triggers:** en aquest apartat es pot triar quina mena d'events volem que activin una build del projecte de forma automàtica.
4. **Build Envirnoment:** triem les accions que volem que el job realitzi abans de començar amb el procés de construcció.
5. **Build:** incloem aquí tots els publishers que volem que s'executin durant el transcurs de la build, en el nostre cas Gradle i Sonar.
6. **Post-build Action:** accions que volem que es realitzin una vegada s'ha construït la build. En aquest apartat podem incloure les notificacions via email sobre el resultat de la build, per exemple.

Tot i que en aquest punt s'ha explicat la creació anticipada dels dos jobs, durant el desenvolupament del projecte s'ha creat primer el job que utilitzarà el Backend primer i fins no començar el desenvolupament de l'aplicació d'Android no se n'ha creat el job.
A partir d'aquest punt comença l'explicació de les tasques com a Dev. Primerament, es parlarà del desenvolupament del Backend, seguit pel Frontend web i per acabar la aplicació Android.

## Developer

### Backend

Abans de començar a parlar del desenvolupament del Backend, fem una ullada al que s'espera que contingui. Per a saber quines tasques s'espera que realitzi el Backend o qualsevol altre element, recordem que estan totes llistades en el Backlog del projecte.

En aquest cas, les tasques relacionades amb el Backend són històries d'usuari les quals haurem de desenvolupar mitjançant BDD. Fent un cop més recordatori del que implica, sabem que:

- S'ha de començar escrivint els diversos escenaris de cada característica amb els steps corresponents.
- Un cop descrits els escenaris s'implementen les funcionalitats descrites.
- Es modifica el codi mentre no passi els tests.

Les tasques que estan implicades en el desenvolupament del Backend són:

| T | Clave | Resumen |
|---|---|---|
| | TQ-21 | As a user I want to monitorize platoons. |
| | TQ-17 | As a user I want to be able to join other user's platoon. |
| | TQ-16 | As a user I want to be able to create platoons. |
| | TQ-15 | As a user I want to make login in the service. |
| | TQ-14 | As a user I want to sign up to the service |

*Imatge 19 User Stories del Backend*

Aquestes històries s'han descompost en tres subtasques que impliquen:

- La escriptura dels escenaris utilitzant Gherkin.
- La implementació de la lògica del servei.
- La implementació dels Step-Definitions i refactorització del codi.

L'ordre de desenvolupament de les diverses característiques ha estat:

1. Implementació del registre.
2. Implementació del login.
3. Implementació de la creació d'escamots.
4. Implementació de l'agregació d'un usuari a un escamot existent.
5. Implementació del sistema de comunicació de coordenades.

A continuació es troba el desenvolupament més detallat de la funcionalitat de registre per a donar exemple de com es realitzarien els diversos passos del desenvolupament en BDD.

Tal com s'ha comentat ja en un parell d'ocasions, el primer que s'han d'escriure són els Features de la característica a desenvolupar. Això implica la definició dels diversos escenaris els quals poden ocórrer quan s'intenti realitzar la acció. Aquests Features segueixen la següent fórmula:

**user_signup.feature** 523 Bytes

```
1  Feature: Sign up into Peloton App
2  As a user I want to be able to signup to be part of the platform
3
4    Scenario: new user signing up
5      Given a new user in "test" "tester" "65100204S" "11/07/1993" "test@test.tt" "testing" "test"
6      When signs up in the service
7      Then the new user is added to the service DB
8
9    Scenario: registered user is signing up
10     Given a registered user in "test" "tester" "65100204S" "11/07/1993" "test@test.tt" "testing" "test"
11     When signs up in the service
12     Then return an error message
```

*Imatge 20 Exemple de feature*

Tal com es pot observar, s'han definit per a la acció de registrar-se dos possibles escenaris:

1. En el primer escenari, es contempla la opció que un usuari nou, és a dir, un que no es troba ja en la nostra base de dades es registri a la aplicació. En tal cas, s'espera que aquest nou usuari s'afegeixi a la base de dades.
2. En el segon, es contempla la possibilitat que un usuari registrat prèviament intenti registrar-se un cop més. En aquest cas, s'espera que l'aplicació respongui amb un missatge d'error.

Un cop definits els diferents escenaris, es comença amb la implementació del codi responsable de fer la funció del registre. El que es vol és que, primerament, es comprovi si l'usuari ja es troba registrat. Segons el resultat d'aquesta comprovació, es vol que, o bé, s'envii un missatge d'error, o bé, es guardi l'usuari nou a la base de dades.

Un cop implementades les funcions necessàries per a que l'usuari es pugui registrar, falta escriure les step-definitions per a aquesta característica. Les step-definitions són la implementació en codi dels steps descrits pels escenaris.

```
@Given("^a new user in \"(.*?)\" \"(.*?)\" \"(.*?)\" \"(.*?)\" \"(.*?)\" \"(.*?)\" \"(.*?)\"$")
public void a_new_user_in(String name, String lastname, String dni, String birthdate, String email,
```

*Imatge 21 Exemple de relació entre feature i step definition*

Com es pot veure en l'anterior figura, l'esmentada anotació @Given entra en acció en el codi per a donar relació entre el Step-Definition i l'escenari descrit amb Gherkin. L'anotació va seguida d'una expressió regular la qual identifica el @Given del codi amb el Given corresponent al step.

En el cas del Given, es pot observar que es passen els paràmetres de l'usuari que es vol utilitzar per al test entre comes dobles flotants. Aquests paràmetres també estan representats en l'expressió regular i en els paràmetres rebuts per la funció.

Què ha de passar en els diferents Steps?
Dir primerament que cada step representa un Given, When o Then de cada escenari. El que s'ha de calcular és el següent:

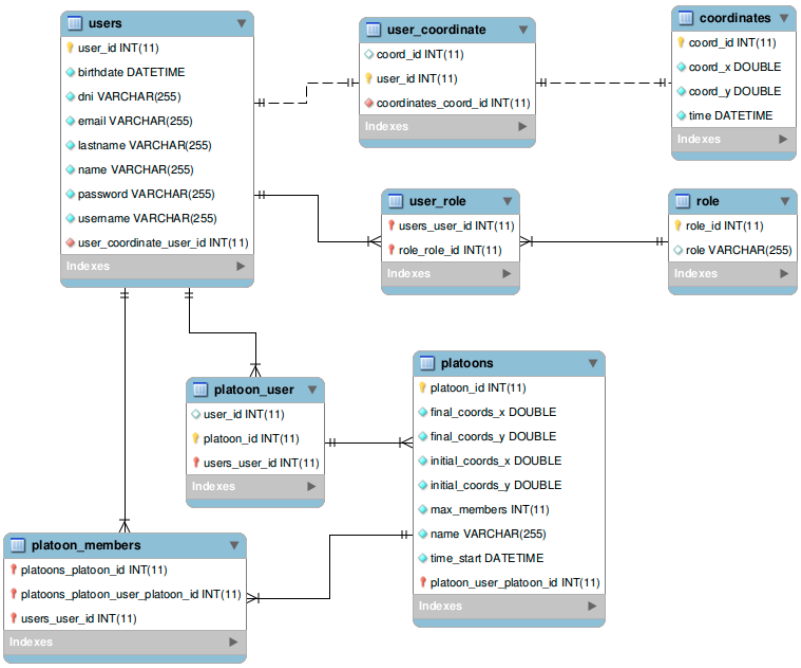
- @**Given:** en aquesta funció hem de comprovar si la condició donada en el Given és correcta o falsa. En el cas d'exemple, comprovarem en base de dades l'existència de l'usuari i en base al retorn de la petició haurem de fer fallar el test.
- @**When:** en aquest step hem de realitzar les crides a la RestAPI per a que realitzi la acció que es vol testejar. En aquest cas, cridarem al registre. En aquest step simplement farem la crida a la funció, no analitzarem el resultat de l'acció.
- @**Then:** en aquest darrer step es comprovarà que el resultat de l'acció realitzat pel @When és l'esperat. Depenent d'aquest resultat es farà que el test falli o passi.

Aquests passos seran els mateixos a seguir per als diferents Features del Backend.

S'han de comentar ara diversos aspectes del Backend: les bases de dades i la seguretat. Per què es parla de bases de dades en plural? Se'n parla en plural perquè en aquest cas concret se n'ha utilitzat dues:

- Una primera base de dades en MySQL la qual es troba allotjada la màquina local on s'executa el Backend la qual conté tota la informació de l'aplicació: usuaris, rols, escamots, coordenades i taules intermèdies.
- Una segona base de dades en H2 la qual es troba en memòria i només existeix quan s'executen els tests i amb aquesta finalitat. Hi ha un script en el projecte el qual dóna les dades inicials necessàries per a que es puguin realitzar tots els tests.

La base de dades MySQL és una base de dades que s'ha definit des de Spring. Spring permet, mitjançant anotacions, la creació de taules a partir de les classes de model definides. Les taules resultants a partir del codi generat amb Spring han donat com a resultat el següent model d'entitat-relació:




*Imatge 22 Diagrama d'entitat relació*

Les decisions preses a l'hora d'implementar les diferents classes de model, les quals han afectat directament al model entitat-relació, han estat:

- Per a cada usuari mantenir seguiment únicament de la seva darrera posició. D'aquesta manera s'aconsegueix una relació 1 a 1 entre usuari i coordenades, doncs la coordenada representarà sempre la última ubicació coneguda de l'usuari.
- Cada usuari tindrà relació de n a m rols. Els rols definits han estat únicament dos, ADMIN i USER. Tot i que no ha estat una característica explotada actualment, la diferenciació de funcions a les que accedir assignant rols als usuaris és una pràctica comuna.
- Entre usuari hi escamot hi haurà dues relacions, una per a definir els membres de l'escamot i l'altra per a definir l'organitzador. D'aquesta manera, tot i que hi haurà una relació n a m en el cas dels membres, hi haurà una relació 1 a n en el cas dels organitzadors.

Per què s'ha de parlar de la seguretat? S'ha de parlar de la seguretat a dos nivells:

1. A nivell de guardar els passwords de l'usuari en la base de dades.
2. A nivell d'accés a les diverses URL de la RestAPI.

El que passa amb el guardar les contrasenyes a base de dades és el fet de guardar informació sensible de l'usuari a la base de dades. El problema es troba en que a la mínima intrusió a la base de dades, l'intrús pot accedir a tota la informació de l'usuari amb la qual podria suplantar la seva identitat o altres.

La solució implementada per a evitar aquest problema ha estat la utilització de BCrypt per a codificar el password de l'usuari abans de guardar-lo en la base de dades. El que fa BCrypt

és convertir el password en un hash el qual comença amb una cadena concreta que indica que és una cadena generada amb BCrypt i el salt fet servir per a calcular-lo, seguit del hash calculat a partir del password.

L'avantatge de guardar un hash és que no es pot tornar a la cadena de la qual procedeix i sempre que es calcula amb la mateixa funció i paraula retorna el mateix valor de hash, de manera que, recalculant el hash i comparant-los, serà suficient per a reconèixer l'usuari i evitar el guardar dades sensibles. Tot i això, BCrypt utilitza salt el qual s'afegeix al càlcul del hash i fa que cada cop que es calculi retorni un valor diferent. Això li atorga un punt extra de seguretat ja que BCrypt és capaç de comparar dos hash i dir si han vingut de la mateixa cadena.

L'altre punt de la seguretat és la transmissió, en la capçalera, d'informació sensible de l'usuari pel camp de Authorization. Les opcions contemplades han estat dos:

- **Basic:** és un sistema d'autenticació senzill el qual consisteix en la concatenació del nom d'usuari i el password mitjançant dos punts ( : ). Un cop concatenats es codifica en Base64 i s'adjunta a la capçalera de Authorization precedit de la paraula Basic i un espai de separació.

```
Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

*Imatge 23 Exemple de Authorization header utilitzant Basic*

- **OAuth2 token:** aquest sistema es basa en la crida al servei de Oauth2 per a obtenir un token. Aquest token conté un access_token, token_type, refresh_token, expires_in i scope. Aquesta tecnologia fa possible accedir a les URL únicament adjuntant a la capçalera el tipus de token seguit de l'access_token en lloc del nom d'usuari i el password. Expires_in indica el temps de validesa del token. Un cop passat aquest temps, es pot utilitzar el refresh_token. Aquest, sol tenir un període d'expiració major al access_token i serveix per a demanar un altre access_token.

```
{
    "access_token": "2f19dba5-d29f-47ed-b5c2-38b2f8f10cbd",
    "token_type": "bearer",
    "refresh_token": "dff8a57e-5399-48df-aaf3-d602050636c6",
    "expires_in": 1199,
    "scope": "read write trust"
}
```

*Imatge 24 Exemple de Token Oauth2*

El primer cas es poc segur degut a que, en el cas que algú estigui atent al transit de la xarxa i obtingui la petició, es pot utilitzar un decodificador de base64 per a desxifrar el nom d'usuari i password que es passa amb la capçalera. A més, aquesta informació sensible s'enviaria amb cada petició per a assegurar que l'usuari està registrat i té permisos per a accedir a les diferents URL.

En el segon cas, es realitza una única petició a la RestAPI en la que es passa el nom d'usuari i password. Les altres peticions es passa en la capçalera d'autorització el token d'accés. El sistema de Backend guarda en memòria els diferents tokens lliurats de manera que pot identificar a quin usuari pertany el token rebut.

Queda un últim punt a comentar respecte el Backend. Aquest punt tracta sobre l'ús de DTOs i DAOs els quals són dos tipus d'objecte utilitzats en diferents processos del funcionament del Backend.

Podem veure que en el projecte hi ha tres tipus de classes diferents a més dels controladors, serveis i classes de configuració. Aquestes són:

- **Classes objecte de negoci:** és un objecte el qual té un conjunt d'atributs, operacions i relacions amb altres objectes del negoci. Contenen dades del negoci i conformen el comportament del negoci.
- **Classes DTO:** un DTO és un objecte de transferència de dades. La seva finalitat és la d'emmagatzemar i transmetre les seves pròpies dades a altres sistemes de l'aplicació. Això vol dir que serà un DTO el tipus d'objecte que s'enviarà als Frontend de l'aplicació, de mateixa manera que l'objecte que rebrà dels Frontend. Poden contenir un subconjunt dels atributs d'un objecte de negoci o fins i tot agrupar-ne més d'un que estigui relacionat.
- **Classes DAO:** un DAO és un objecte d'accés a dades. Ens ofereix una interfície entre l'aplicació i un dispositiu d'emmagatzematge de dades, una base de dades en aquest cas. Això vol dir que proporcionarà a les altres classes del Backend les diverses funcions per a realitzar peticions a la base de dades. Seran els únics objectes que realitzaran aquest tipus de crides i s'associaran a un objecte del negoci concret.

Les respostes del Backend consistiran en un objecte el qual serà un ResponseDTO. Aquest contindrà, segons la resposta que s'envii:

- Un enter que serà el codi de la resposta.
- Un String per a descriure l'estatus de la resposta.
- Un objecte que correspondrà al DTO de l'objecte del model que s'envii.
- Un DTO per a l'error si es dóna el cas. Aquest contindrà un codi d'error i una descripció de l'error.

El DAO és, per altra banda, un component que subministra una interfície entre la aplicació i un dispositiu d'emmagatzematge de dades, això vol dir que el DAO és l'encarregat de proporcionar al Backend les peticions a base de dades.

## Web

La realització del Frontend inclou el desenvolupament d'una pàgina web utilitzant Angular 5. Per a la implementació de la pàgina s'ha utilitzat doncs HTML, CSS i TypeScript. Tot el necessari per a realitzar la web s'ha instal·lat utilitzant NodeJs, amb l'ajuda del qual s'instal·la Angular CLI.

La creació del projecte i el desplegament en local de la web ha estat tant senzill com utilitzar les següents instruccions:

```
>  ng new my-dream-app

>  cd my-dream-app

>  ng serve
```

*Imatge 25 Comandes d'inici d'un projecte Angular*

L'ús d'aquestes instruccions implica:

1. La generació d'una aplicació preparada per a utilitzar d'Angular.
2. Entrar a la carpeta que conté l'aplicació.
3. Desplega l'aplicació en local de manera que qualsevol canvi efectuat en l'aplicació ho refresca automàticament per a veure els canvis realitzats mentre es desenvolupa.

Tenint en compte que la nostra aplicació utilitzarà una RestAPI que s'executarà en un altre port per a realitzar tots els càlculs, s'haurà d'utilitzar la tercera instrucció modificada amb la finalitat que el programa al executar-se accepti CORS (cross-origin resource sharing). La solució està en configurar un petit proxy:

```
{
    "/api": {
        "target": "http://localhost:8080",
        "pathRewrite": {"^/api" : ""},
        "secure": false
    }
}
```

*Imatge 26 Exemple de fitxer proxy.conf.json*

Amb aquest Json li estem dient que quan a la ruta que cridi es trobi la cadena /api ho canvii per la ubicació del Backend, en el meu cas localhost:8080. La crida al ng serve es farà doncs de la següent manera:

```
$ ng serve --proxy-config proxy.conf.json
```

*Imatge 27 Comanda d'execució Angular amb proxy*

Un cop iniciada una aplicació, les tasques que s'hauran d'implementar són les següents:

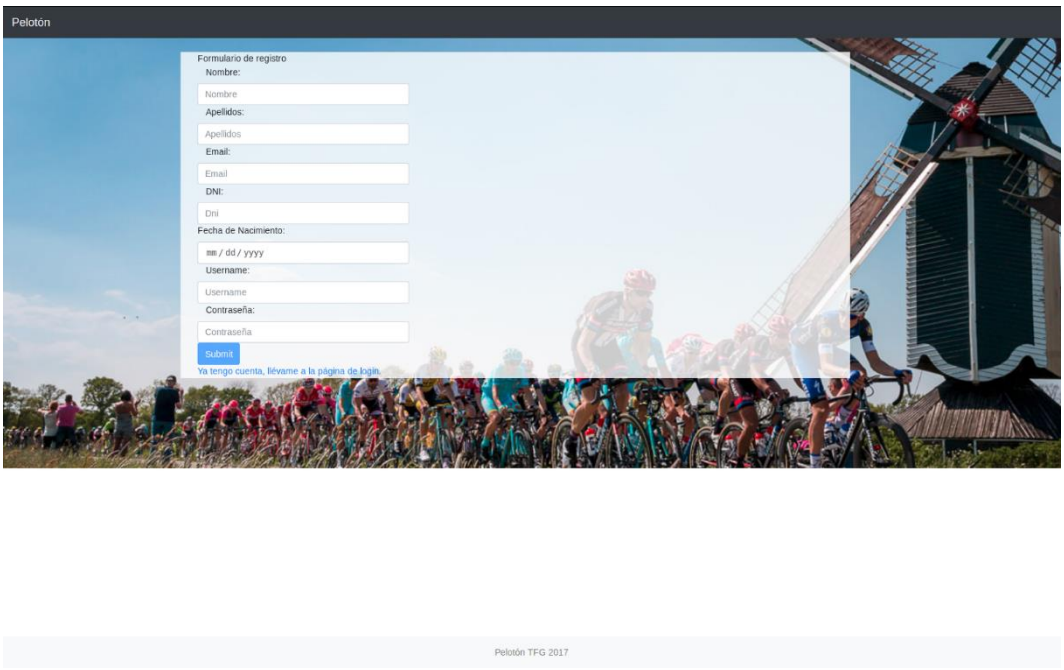

*Imatge 28 Llista de tasuqes de la web*

Aquestes tasques són també tasques extretes de l'anàlisi de les èpiques del Backlog i impliquen la següent feina i resultats:

- Una pàgina que inclogui un formulari per a introduir les dades per a que l'usuari es pugui registrar.
- Una pàgina que inclogui un formulari per a introduir les dades per a que l'usuari pugui fer el login.
- Una pàgina que sigui la pàgina d'inici en la que es mostri el llistat d'escamots existents i es doni l'opció a l'usuari de crear-ne de nous.
- Una pàgina amb el formulari per a que l'usuari pugui crear els escamots nous.
- Una pàgina que mostri els detalls d'un escamot seleccionat per a que l'usuari pugui unir-se o per a que pugui seguir tots els membres en el seu recorregut o a un membre en concret.

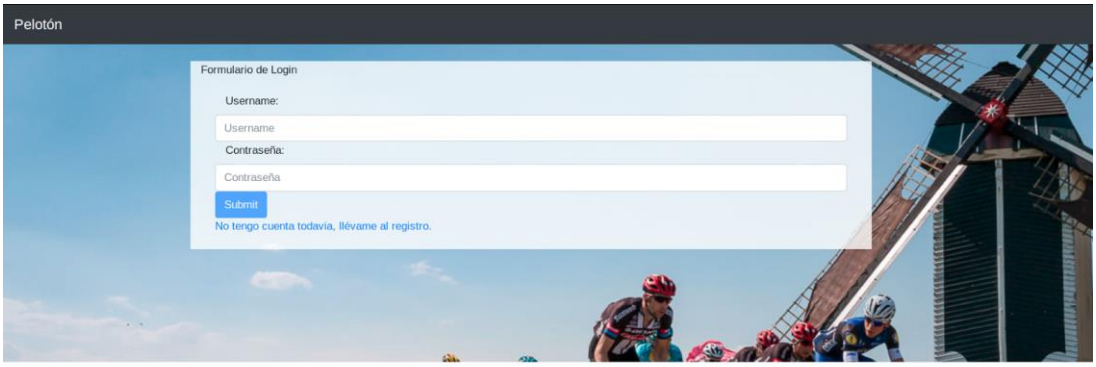
Com es pot veure, aquestes tasques del Backlog són lo suficient generals per a que es puguin esmicolar encara més amb la finalitat de que cada tasca sigui tant específica com sigui possible.

En el registre es demanen les dades bàsiques per a poder guardar l'usuari definit en el model. En cada camp del formulari es comprova que les dades introduïdes compleixin unes especificacions descrites amb una RegEx i no deixa que s'envii a no ser que el formulari sigui vàlid, és a dir, es compleixin totes les especificacions donades per a cada camp.




*Imatge 29 Pàgina de registre*

En la pàgina de login trobem dos camps, l'un per a que l'usuari posi el username i l'altre per a que hi posi el password. També hi ha un control per a aquests camps el qual correspon amb els paràmetres d'acceptació utilitzats en el registre. Si aquests camps no passen aquesta acceptació, no es pot enviar el formulari. A més conté un missatge d'error en cas que el login no es realitzi amb èxit.

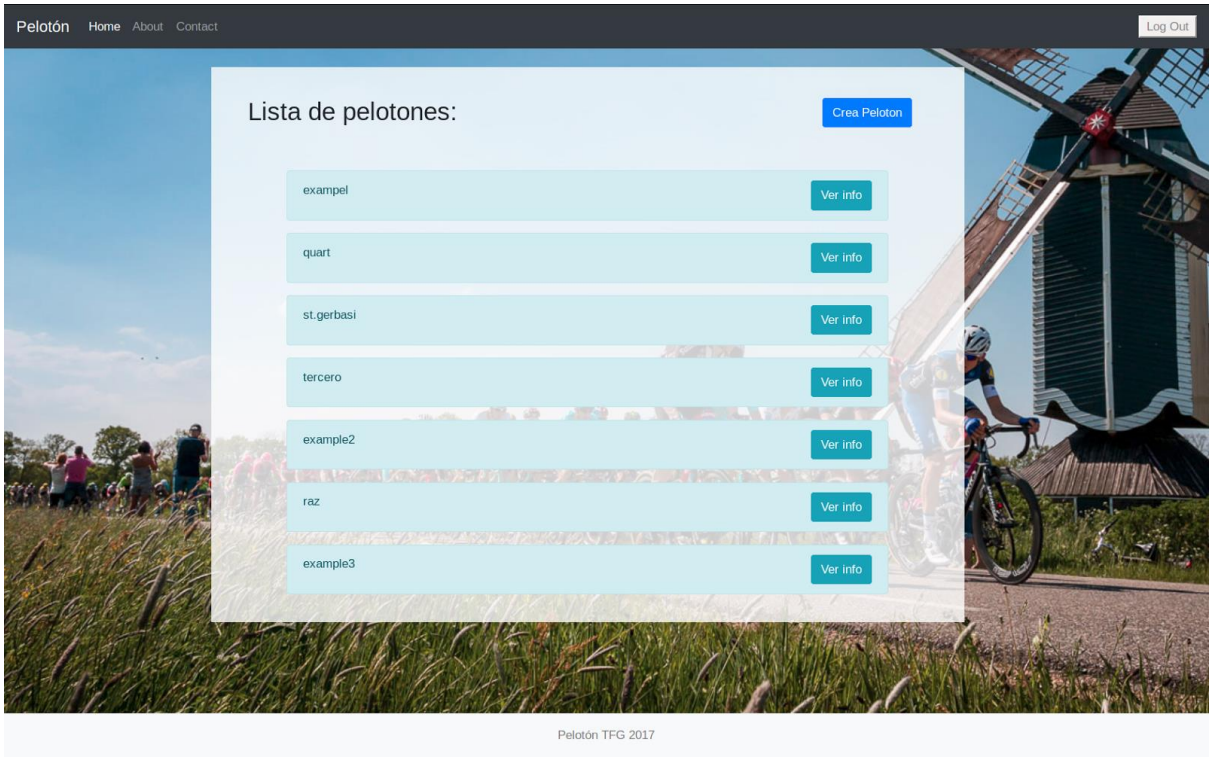



*Imatge 30 Pàgina de login*

Un cop passat pel login, es guarda el token en el LocalStorage del navegador i es pot obtenir d'allà mateix a l'hora de realitzar les peticions al Backend. Després, l'usuari va a petar a la pàgina d'inici o home.

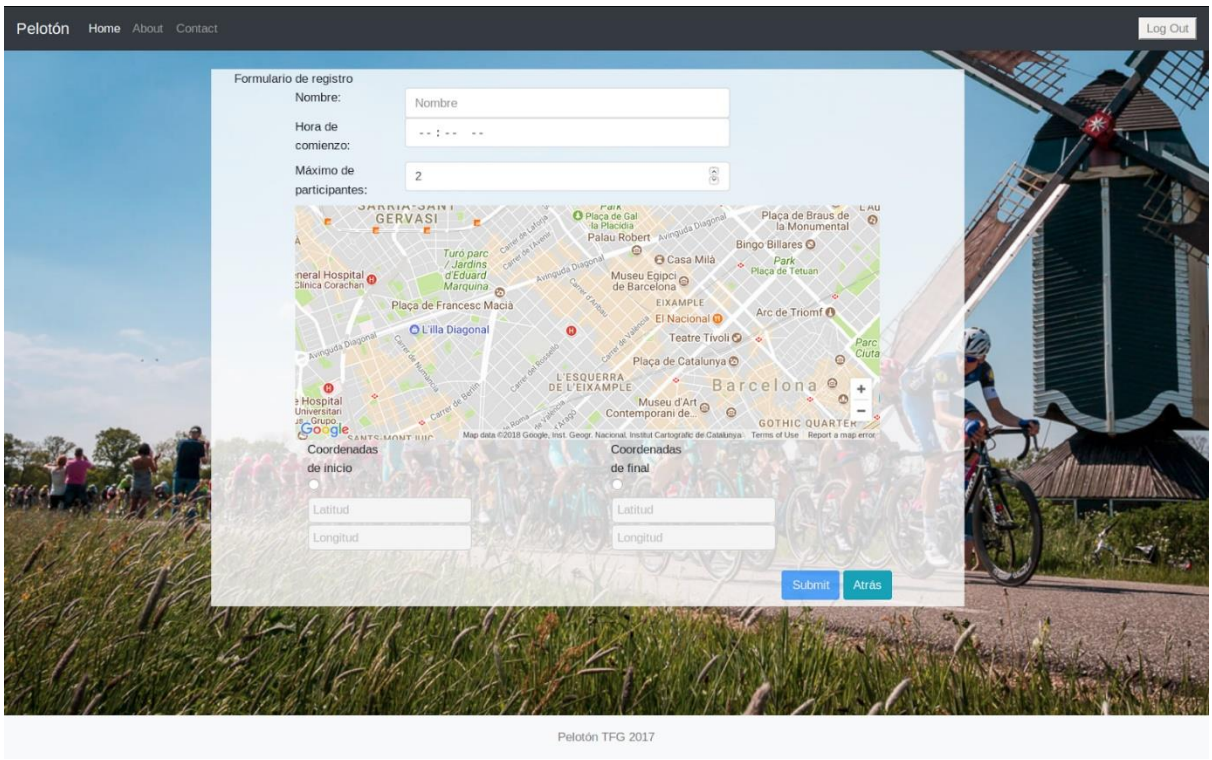
En aquesta pàgina, li apareix a l'usuari un botó per a poder sortir de la plataforma fent un logout. Això el retornaria a la pàgina de login.

En la pàgina del home, l'usuari pot veure un llistat de tots els escamots que ell i altres usuaris han registrat i pot crear-ne un. L'altre opció que té l'usuari en aquesta pàgina és la de visualitzar al detall els diferents escamots llistats.




*Imatge 31 Pàgina home*

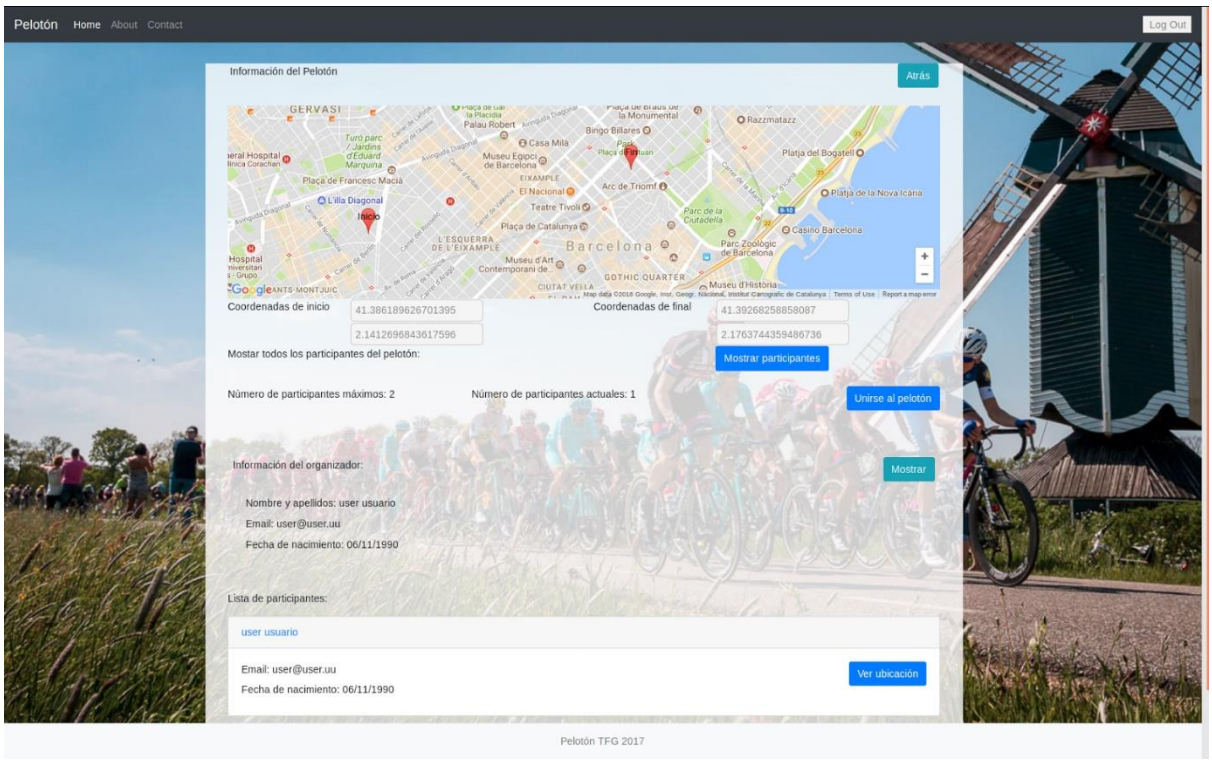
En la creació d'un escamot, l'usuari es troba amb un mapa de l'API de Google, el qual està centrat a Barcelona, en el que ha de seleccionar dues ubicacions clicant-hi al damunt. Aquestes ubicacions reflecteixen la posició d'inici i fi de la ruta de l'escamot i queden marcades amb un Marker al mapa i poden ser modificades canviant l'opció de seleccionar una coordenada o l'altra. Els altres camps també tenen un sistema de validació i un cop el formulari queda omplert, es pot enviar i queda guardat a la base de dades.




*Imatge 32 Pàgina de creació d'escamot*

En la pàgina de detalls de l'escamot, l'usuari pot veure la informació de l'escamot la qual inclou:

- El nom donat a l'escamot.
- La hora d'inici de l'activitat de l'escamot.
- El mapa amb els marcadors d'inici i fi de ruta.
- El nombre de places màximes de l'escamot.
- El nombre de places actuals ocupades de l'escamot.
- Informació bàsica sobre l'organitzador de l'escamot. L'organitzador és aquell usuari el qual ha creat l'escamot. Aquesta informació inclou el seu nom i el correu electrònic.
- Mostra també el llistat d'usuaris que estan inscrits en l'escamot.
- Dóna també la opció de mostrar al mapa amb uns marcadors la última ubicació registrada de cadascun dels membres de l'escamot. Aquesta informació la pot donar d'un únic membre o de tots els membres de l'escamot alhora.




*Imatge 33 Pàgina de detalls d'escamot*

Un cop desenvolupades les diverses pàgines les quals conformaran la web, hem de trobar un mecanisme el qual ens garanteixi que únicament aquells usuaris que presentin un token d'accés podran veure les pàgines que no siguin ni el login ni el registre.

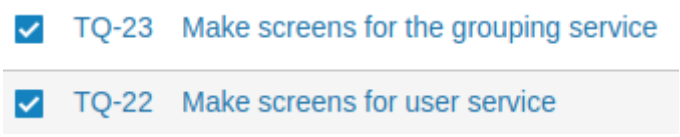
Tal mecanisme es troba definit en una classe dins la web que s'anomena interceptor. Aquest, com el seu nom indica, s'encarrega d'interceptar totes les peticions que realitza l'usuari. Realitza dues tasques:

1. Comprova si la petició que ha interceptat té un paràmetre Authorization a la capçalera i, de no tenir-lo, l'afegeix ell. S'utilitza per a afegir el token d'accés a cada petició de forma automàtica.
2. Mira si hi ha hagut un error d'autorització en la resposta del Backend. De ser així, s'esborra el possible token existent i es redirigeix a l'usuari a la pàgina d'inici de sessió.

## Aplicació Android

L'aplicació per a Android serà una aplicació desenvolupada utilitzant Android Studio. Aquesta, ha estat desenvolupada en Java utilitzant un parell extra de dependències, una anomenada Dagger i l'altra anomenada Butterknife. Aquestes dues dependències extres serveixen per a dotar a Android Studio de la capacitat d'injectar codi i poder-se referir a components de la vista mitjançant anotacions.

En aquesta darrera part de la implementació, la feina a realitzar extreta del Backlog queda reflectida en aquestes dues tasques:

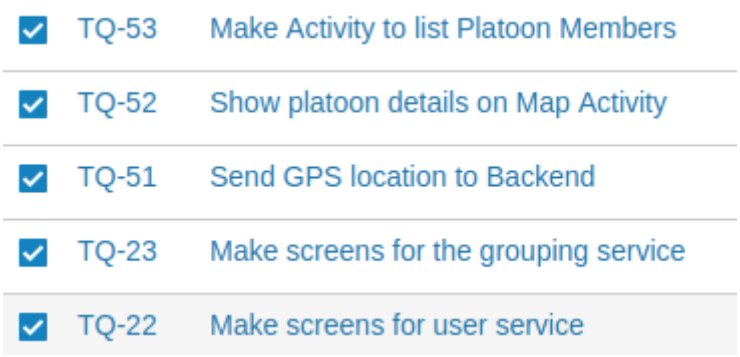



*Imatge 34 Llista de tasques d'Android*

Segons aquesta definició de tasques i el que s'hauria de fer, aquestes tasques impliquen:

- Crear una Activity en la que l'usuari pugui fer el login.
- Crear una Activity en la que un nou usuari es vulgui registrar a la plataforma.
- Crear una Activity, basat en l'aplicació web, que sigui la pantalla de home on es llistin els diversos escamots.
- Crear una Activity en la que es mostrin els diversos detalls de l'escamot i es doni la opció per a que l'usuari s'hi uneixi.
- Crear una opció per a que l'usuari pugui enviar o deixar d'enviar les dades d'ubicació al Backend.

Un cop més les tasques descrites al Backlog són massa generalitzades i s'haurien d'esmicolar en tasques més petites i específiques. En aquest cas, les tasques que van acabar sorgint de l'anàlisi d'aquestes tasques són les següents:




*Imatge 35 Llista final de tasques d'Android*

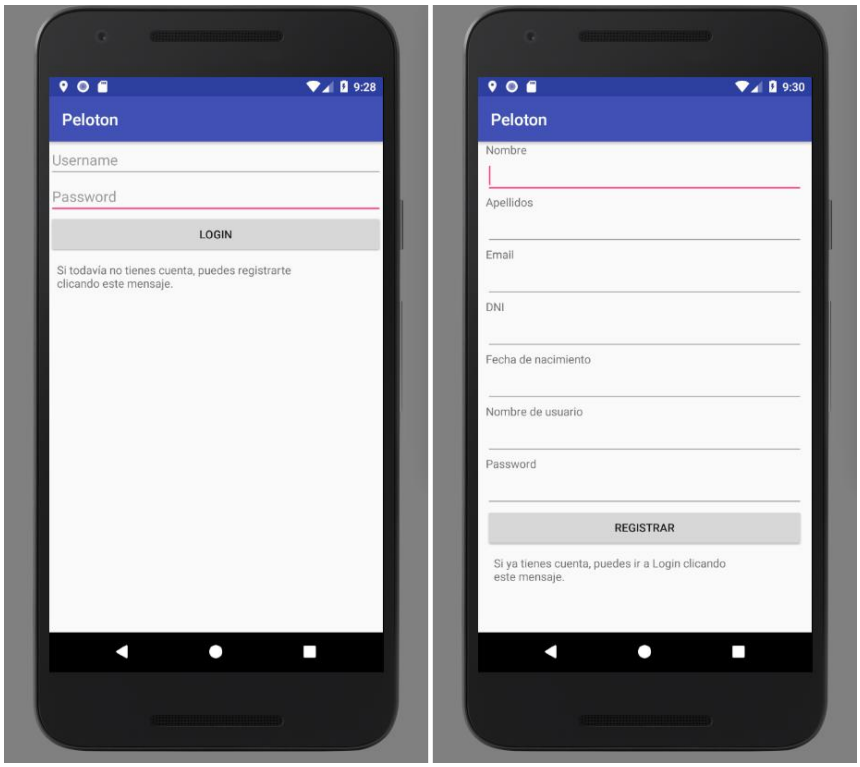
Si bé és cert que es podria millorar algun aspecte extra del nou Backlog, han sorgit algunes tasques específiques que abans no estaven ni esmentades. Així doncs, tenim que:

- TQ-22 ens implica crear login, registre i un home.
- TQ-23 suposa crear la llista d'escamots al home i els detalls de l'escamot.
- TQ-51 implica afegir la funcionalitat d'enviar les coordenades.
- TQ-52 implica mostrar en un mapa la informació de l'escamot.
- TQ-53 implica mostrar la llista de membres de l'escamot.

L'usuari es trobarà que l'aplicació s'inicia mostrant la pantalla de login. En aquesta pantalla se li apareix un parell de camps de text en els que pot introduir el nom d'usuari i el password per a entrar a l'aplicació. Qualsevol error degut a que l'intent hagi estat fallit, se li mostrarà

per pantalla en un Toast. L'altre detall observable de la pantalla de login és un text el qual, de ser clicat, et porta a la pantalla de registre.

En la pantalla de registre també hi ha un formulari amb els camps necessaris per a la introducció de totes les dades necessàries per a la creació d'un nou usuari i també un text el qual permet a l'usuari tornar a la pantalla de login donat el cas que ja tingui compte. S'ha de dir que qualsevol intent de l'usuari de tornar a la pantalla anterior pot ser realitzat apretant el botó de navegació propi del telèfon.

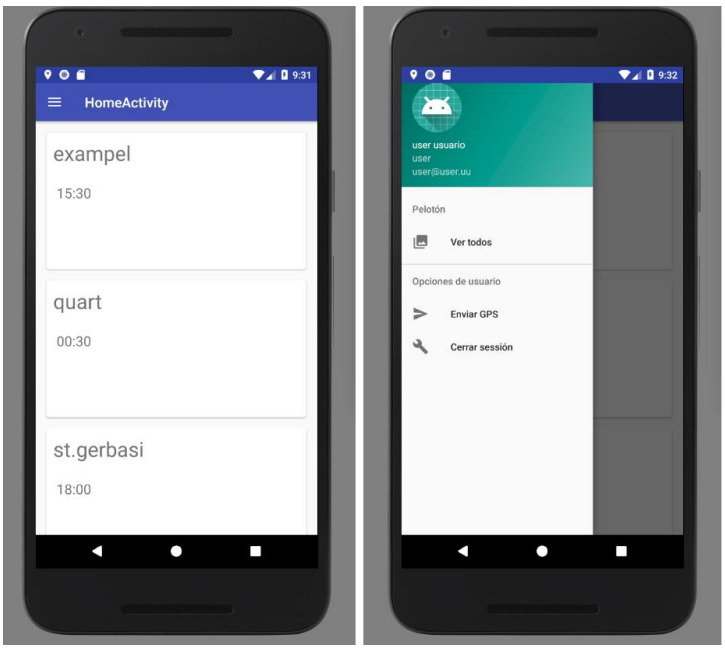



*Imatge 36 Activities de login i registre*

Un cop l'usuari ha realitzat el login, va a parar a la pantalla del home. Aquí se li mostrarà el llistat de tots els escamots creats tant per ell com per la resta d'usuaris. A més, a la capçalera es troba una icona que permet a l'usuari desplegar un menú lateral.

En aquest menú lateral l'usuari podrà seleccionar la opció d'enviar les dades d'ubicació al Backend per a que actualitzi la última coordenada coneguda de l'usuari i la opció per a fer un logout.

En aquesta pantalla, l'usuari pot clicar també dues vegades el botó del telèfon per a anar enrere i també realitzaria d'aquesta manera el logout.




*Imatge 37 Fragments home i menú lateral*

L'altra opció que es té des de la pantalla del home és la de clicar a un dels escamots llistats. Això farà que se li mostri a l'usuari la pantalla dels detalls de l'escamot. En aquesta pantalla, l'usuari pot veure la informació de l'escamot presentada en camps de text. A més d'aquests camps de text, l'usuari pot veure tres botons en pantalla:

- Un primer botó que permet veure una nova pantalla amb un mapa, la qual mostra els dos marcadors d'inici i fi de la ruta.
- Un botó que permet veure una nova llista en pantalla la qual conté dades bàsiques de cada membre inscrit a l'escamot.
- Un darrer botó que dóna a l'usuari la opció de unir-se a l'escamot.


*Imatge 38 Activity de detalls de l'escamot*

# Conclusió

A partir del treball realitzat s'ha aconseguit tenir desenvolupades les diverses parts proposades en el plantejament del projecte, és a dir, el Backend, la web i l'aplicació per a dispositius Android, a més dels serveis utilitzats per a la IC.

El Backend realitza les diverses tasques assenyalades al principi a més d'algunes altres que s'han previst durant el desenvolupament de la web o l'aplicació. Algunes d'aquestes funcions han estat:

- La recuperació del perfil de l'usuari un cop ha fet el login i ha entrat al home del Frontend.
- L'obtenció dels diferents escamots creats per tots els usuaris per a llistar-los en el home.

Tot i que amb la previsió inicial no ho contemplava, el desenvolupament de la web va fer-me adonar de la necessitat d'una pàgina a la que l'usuari anés a parar un cop hagués fet el login. La pantalla intermitja resultant ha estat la del home i ha resultat idònia per a mostrar els escamots i donar la possibilitat de crear-ne de nous.

Per la banda de l'aplicació Android ja anava amb l'estimació d'haver de fer com a mínim les mateixes o més pantalles. Aquí, en topar amb una pantalla més petita, com és la d'un mòbil comparada amb la d'un ordinador, sorgeix la decisió d'afegir més pantalles, en aquest cas, a la informació de l'escamot.

En la informació de l'escamot surt únicament la informació de l'escamot i l'organitzador i la resta d'informació es reserva per a si s'utilitza algun dels botons presents en la pantalla.

Així doncs, tot i que els requeriments funcionals no han canviat durant el transcurs del projecte, es pot apreciar que efectivament hi ha hagut un canvi en les necessitats a l'hora de desenvolupar en una plataforma diferent, ja fos incloent funcions al Backend, que no s'havien esperat o considerat de bon principi, o afegint pantalles intermèdies per a mostrar certa informació a l'usuari.

El que intento destacar és el canvi. Un canvi que s'ha fet present al llarg del transcurs del treball i que implica un anàlisi nou o la contemplació de noves opcions o aproximaments a la solució que originalment s'espera. El canvi, i l'adaptació que suposa, m'ha permès el compliment del que m'havia proposat desenvolupar i l'he acabat considerant útil i necessari en el desenvolupament de Software.

Els objectius del projecte, els quals engloben el desenvolupament de l'aplicació i del desplegament dels serveis de integració contínua, s'han anat complint al llarg dels diversos Sprints realitzats setmanalment. Han estat 10 Sprints en total en els quals he:

- Après a configurar i utilitzar tecnologies, les quals desconeixia, per a la utilització d'integració contínua al llarg d'un projecte.
- Aprés a utilitzar Angular 5 per al desenvolupament d'una aplicació web.
- Refrescat coneixements sobre el desenvolupament d'aplicacions per a Android.
- Refrescar i ampliar coneixements sobre Java i RestAPIs.

No considero que hagi deixat cap objectiu dels proposats al principi del treball irresolt, tot i que és cert que hi ha diversos aspectes del projecte els quals voldria aprofundir.

Un d'ells és la dificultat que ofereix Angular 5 a la utilització de Bootstrap. Bootstrap és una eina bàsica i clàssica que es sol incloure en el desenvolupament web per a fer més atractiva la pàgina a ulls de l'usuari, tot i això, he vist que hi ha dos alternatives que es complementen per a afegir les diverses característiques que Bootstrap ofereix:

- **Angular Flex**: afegeix al projecte la possibilitat d'organitzar els elements del layout en columnes i la seva distribució dins la columna.
- **Angular Materials:** afegeix un llistat de components els quals poden ser utilitzats i són més vistosos que els definits per defecte amb HTML.

L'altre aspecte en el que voldria aprofundir és en el desenvolupament d'aplicacions Android. Voldria aprendre millor com utilitzar i configurar les eines Dagger i Butterknife que he utilitzat per a fer l'aplicació o el Retrofit utilitzat per a les connexions entre l'aplicació i el Backend. Android també ofereix un bon ventall de components el qual no he utilitzat i aspectes dels utilitzats, han estat poc explotats i també en voldria saber més.

## Treball futur

El treball futur que deixa aquest treball es pot dividir en dues parts: una de millores sobre el que ja existeix i l'altra de noves funcionalitats.

En quant a les millores que se'm passen pel cap un cop considerat acabat el desenvolupament, el que proposo és:

1. Canviar la web per a que en lloc d'intentar utilitzar Bootstrap utilitzi Angular Flex i Angular Materials. Aquesta millora intentaria solucionar els problemes i dificultats sorgits per la incompleta compatibilitat entre Bootstrap i Angular 5.
2. Canviar els botons per a que es mostrin els membres de l'escamot per uns botons Toggle que donin informació visual de si es troba clicat o no. D'aquesta manera es milloraria el feedback obtingut per l'usuari.
3. Mirar com parar la crida automàtica a la funció que va actualitzant la ubicació dels usuaris en pantalla en la web. Aquesta funció es segueix cridant fins que no es desmarca clicant de nou a mostrar un usuari o es canvia per a mostrar l'escamot. En cas que es canvii de URL aquesta funció es segueix cridant i no hauria de passar.
4. Canviar les Activities de l'aplicació Android relacionades amb els detalls de l'escamot per Fragments. D'aquesta manera es pot aprofitar el menú lateral que apareix al Home en les altres Activities.

Per altra banda, les noves funcionalitats amb les que ampliaria el treball són les que segueixen:

1. La possibilitat de poder veure els escamots propis escamots. Aquesta opció o alguna variant per a poder veure uns escamots concrets de forma més ràpida.
2. La possibilitat de veure l'ubicació d'un membre de l'escamot des de l'aplicació Android. D'aquesta manera es podria veure la ubicació d'algú que s'hagués quedat enrere per a que el grup l'esperi o sàpiga com a mínim on és per si l'hi ha passat alguna cosa.
3. La possibilitat de canviar l'estat de l'escamot. D'aquesta manera se li podria donar inici o fi a l'activitat o fins i tot actualitzar-ne paràmetres.
4. La possibilitat d'editar la visibilitat de l'escamot a tot el públic o únicament a l'organitzador. D'aquesta manera es dotaria als usuaris de més privacitat.

# Bibliografia


1. (GitLab) Omnibus GitLab documentation-GitLab Documentation [20 Octubre 2017]
   Disponible: https://docs.gitlab.com/omnibus/README.html
2. (Gitlab) SMTP settings [8 Gener 2018]
   Disponible: https://docs.gitlab.com/omnibus/settings/smtp.html
3. (Jenkins) Getting Started with the Guided Tour [23 Octubre 2017]
   Disponible: https://jenkins.io/doc/pipeline/tour/getting-started/
4. (Jenkins) Terminology [11 Gener 2018]
   Disponible: https://wiki.jenkins.io/display/JENKINS/Terminology
5. (Sonar) SonarQube: instalación y configuración [24 Octubre 2017]
   Disponible: http://enrikusblog.com/sonarqube-instalacion-y-configuracion/
6. (Spring) Why Spring Boot [5 Gener 2018]
   Disponible: https://dzone.com/articles/why-springboot
7. (Spring) Spring Boot - Simplifying Spring for Everyone [5 Gener 2018]
   Disponible: https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone/
8. (Spring) Enabling Cross Origin Requests for a RESTful Web Service [27 Nov 2017]
   Disponible: https://spring.io/guides/gs/rest-service-cors/
9. (Spring) Setting Up Swagger 2 [28 Novembre 2017]
   Disponible: http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api
10. (Spring) Retrieve user information in Spring Security [15 Novembre 2017]
    Disponible: http://www.baeldung.com/get-user-in-spring-security
11. (Spring) Spring REST Api + OAuth2 + AngularJs [16 Novembre 2017]
    Disponible: http://www.baeldung.com/rest-api-spring-oauth2-angularjs
12. (Spring) Spring Boot and Oauth2 [10 Novembre 2017]
    Disponible: https://spring.io/guides/tutorials/spring-boot-oauth2/
13. (Spring) Registration with Spting Security - Password Encoding [10 Nov 2017]
    Disponible: http://www.baeldung.com/spring-security-registration-password-encoding-bcrypt
14. (Spring) Getting startet with Spring Data JPA [9 Novembre 2017]
    Disponible: https://spring.io/blog/2011/02/10/getting-started-with-spring-data-jpa/
15. (Angular) Angular CLI [22 Novembre 2017]
    Disponible: https://cli.angular.io/
16. (Angular) Angular [22 Novembre 2017]
    Disponible: https://angular.io/
17. (Angular) Angular QuickStart [22 Novembre 2017]
    Disponible: https://angular.io/guide/quickstart
18. (Angular) Bootstrap 4 components, powered by Angular [ 3 Decembre 2017]
    Disponible: https://ng-bootstrap.github.io/#/home
19. (Angular) AGM - Angular Google Maps [5 Decembre 2017]
    Disponible: https://angular-maps.com/api-docs/agm-core/index.html

20. (Angular) Getting started | Angular Material [17 Gener 2018]
Disponible: https://material.angular.io/
21. (Angular) Provides HTML UI layout for Angular applications [17 Gener 2018]
Disponible: https://github.com/angular/flex-layout
22. (Android) User Interface [10 Decembre 2017]
Disponible: https://developer.android.com/guide/topics/ui/index.html
23. (Android) Dialogs [13 Decembre 2017]
Disponible: https://developer.android.com/guide/topics/ui/dialogs.html
24. (Android) Toasts [13 Decembre 2017]
Disponible: https://developer.android.com/guide/topics/ui/notifiers/toasts.html
25. (Android) Fragments [20 Decembre 2017]
Disponible: https://developer.android.com/guide/components/fragments.html
26. (Android) Request Permissions at Run Time [28 Decembre 2017]
Disponible: https://developer.android.com/training/permissions/requesting.html
27. (Android) Enviando Datos con el Cliente HTTP Retrofit 2 [12 Decembre 2017]
Disponible: https://code.tutsplus.com/es/tutorials/sending-data-with-retrofit-2-http-client-for-android--cms-27845
28. (Android) Using Retrofit 2.x as REST client - Tutorial [13 Decembre 2017]
Disponible: http://www.vogella.com/tutorials/Retrofit/article.html
29. (Android) How to clear all activity stack in Android [20 Decembre 2017]
Disponible: https://tips.androidhive.info/2013/10/how-to-clear-all-activity-stack-in-android/
30. (BDD) BDD en Java con Cucumber [24 Octubre 2017]
Disponible: http://www.notodocodigo.com/blog/bdd-behaviour-driven-development-en-java-con-cucumber/
31. (BDD) BDD, Cucumber y Gherkin [2 Gener 2018]
Disponible: https://www.genbetadev.com/metodologias-de-programacion/bdd-cucumber-y-gherkin-desarrollo-dirigido-por-comportamiento
32. (JIRA) JIRA [3 Gener 2018]
Disponible: https://es.wikipedia.org/wiki/JIRA
33. (Agile) Desarrollo ágil de software [4 Gener 2018]
Disponible: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
34. (Agile) La historia de usuario no es el “requisito” de las metodologías ágiles [4 Gener 2018] Disponible: http://www.javiergarzas.com/2011/12/historia-de-usuario-diferente-de-requisito.html
35. (Memòria) Objetos de negocio [18 Gener 2018]
Disponible: https://www.ibm.com/support/knowledgecenter/es/SSLKT6_7.6.0/com.ibm.mt.doc/configur/c_bos.html